TEKNILLINEN KORKEAKOULU
TIETOTEKNIIKAN TALON KIRJASTO
KONEMIEHENTIE 2
02150 ESPOO

TEKNILLINEN KORKEAKOULU
TIETOJENKÄSITTELYOPIN
KÄSIKIRJASTO

TEKNILLISEN KORKEAKOULUN TIETOTEKNIIKAN OSASTO

Sumean logiikan soveltaminen sähkönjakeluverkon tietojärjestelmässä

Jarmo Korhonen

Työn valvoja:
Heikki Saikkonen

Työn ohjaaja:
Harri Salmivaara

DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Jarmo Korhonen
Työn nimi: Sumean logiikan soveltaminen sähkönjakeluverkon tietojäjrestelmässä
Päivämäärä: 18.11.1995
Sivumäärä: 70

Osasto: Tietotekniikan osasto/Tietojenkäsittelytekniikan laitos
Professuuri: Ohjelmistojärjestelmät/Tik-76

Työn valvoja: Heikki Saikkonen
Työn ohjaaja: Harri Salmivaara

Tässä työssä tarkastellaan mahdollisuuksia lisätä tietojärjestelmiin älykkyyttä sumean logiikan avulla. Aihetta on lähestytty käytännön esimerkin kautta, varmistaen että jotain toimivaa ja hyödyllistä on mahdollista saada aikaan. Sovellusesimerkkinä on sähkönjakeluverkon tietojärjestelmä. Sen kunnossapitosovellukseen määritellään sumeaa logiikkaa hyödyntävä laajennus. Työssä on myös määritelty yleisemmin sähkönjakeluverkon tietojärjestelmän problematiikkaan liittyen sumean logiikan säännöstö paikkatiedonhallintaan.

Työn sisältö jakautuu kolmeen osaan:

Ensimmäisessä osassa esitetään sähkönjakeluverkon kunnossapidon nykyiset käytännöt. Verkkojen kunnossapidosta ei ole olemassa kirjallisuutta, joten kuvatut periaatteet ovat suurimmaksi osaksi synteesi eri laitosten omista sisäisistä käytännöistä. Nykyisen käytännön kuvaus on samalla tarvemäärittely kunnossapidon sumealle järjestelmälle. Ensimmäisessä osassa kuvataan myös sumean logiikan teoria tässä työssä tarvittavilta osin.

Toisessa osassa mallitetaan kunnossapidon käytännöt tietojärjestelmäksi ja kuvataan esimerkinomaisesti kuinka sumean logiikan säännöillä voidaan mallittaa kunnossapidon aiempia käytäntöjä. Sumean logiikan osamodulin toteutus suunnitellaan tässä työssä käytetyiltä osin. Kunnossapitosovelluksen toteutusta suunnitellaan niiltä osin kuin se liittyy sumean logiikan käyttöön ja yleisiin rakenteisiin, menemättä tarpeettomiin yksityiskohtiin.

Kolmannessa osassa määritellään paikkatiedonhallinnan sumeita sääntöjä niiltä osin kuin ne ovat hyödynnettävissä sähkönjakeluverkkojen paikkatietojärjestelmässä ja kunnossapidossa. Paikkatiedonhallinnan sääntöjen toteutusta on tutkittu periaatteellisella tasolla, lähinnä mitä laajennuksia se vaatii yleiseen sumean logiikan käsittelyyn.

Työssä on esitetty algoritmit ja tietorakenteet periaatteellisella tasolla. Ohjelmaesimerkkejä on käytetty rajallisesti lähinnä selvittämään rakenteita. Toteutuksen yksityiskohtiin ei mennä lainkaan.

Avainsanat: Sähkönjakeluverkko, kunnossapito, sumea logiikka, paikkatiedonhallinta

Tämän työn syntymisessä tärkeä osuus oli sillä asiantuntemuksella jota sain sähköalan ihmisiltä. Sitä tietoa ei kirjoista löytynyt. Kiitän myös Tekla Oy:tä ja Harri Salmivaaraa kärsivällisyydestä ja tuesta.

Jarmo Korhonen
27.11.1995

Sisällysluettelo

1 Johdanto

Sumeaa logiikkaa on sovellettu laajalti teollisuuden säätölaitteissa ja automaatiojärjestelmissä. Sen sijaan ohjelmistojärjestelmissä sen käyttö on ollut vähäistä, mistä on osoituksena myös aihetta käsittelevän kirjallisuuden vähäisyys. Kirjallisuudessa sumeaa logiikkaa on käsitelty lähinnä matematiikan ja automaation näkökulmasta, ohjelmistotekniikan jäädessä sivuosaan. Tämä työ liittyy löyhästi VTT:n älykkäiden järjestelmien tutkimusprojektiin, jossa toimeksiantaja on yrityspartnerina.

Tämän työn tavoitteena on selvittää, tarjoaako sumea logiikka mahdollisuuksia lisätä ohjelmistojärjestelmiin sellaista älykkyyttä joka vähentää käyttäjän työtaakkaa tehtävissä jotka on aiemmin mielletty kokonaan ihmisten hoidettaviksi. Tutkimuskenttä on tavattoman laaja, mutta tässä työssä on rajoituttu ainoastaan osaan siitä. Monet kehitysnäkymät on siirretty mahdollisten jatkotutkimusprojektien puolelle. Tätä työtä voi jossain määrin pitää myös toteutettavuustutkimuksena (feasibility study) koska aihetta on lähestytty käytännön esimerkin kautta, varmistaen että jotain toimivaa ja hyödyllistä on mahdollista saada aikaan.

Aiheen käsittely tapahtuu lähtien käytännön sovellustarpeesta. Sähkönjakelulaitokset käyttävät verkkojensa hallintaan Xpower-verkkotietojärjestelmää, jonka avulla ne hoitavat verkon suunnittelun, dokumentoinnin ja reaaliaikaisen hallinnan. Järjestelmän avulla on mahdollista hoitaa myös kunnossapitoon ja tarkastustoimintaan liittyviä tehtäviä, mutta osa aikataulu-, resurssi-, yms. tekijät jäävät nykyisellään käyttäjien hoidettaviksi; järjestelmän avulla hallitaan kunnossapidon tarpeessa olevat kohteet ja niihin liittyvä tietous. Konkreettisena tavoitteena on tutkia onko mahdollista tuottaa sumean logiikan avulla sellainen kunnossapidon järjestelmä, joka tuottaa automaattisesti arvion kunnossapidon tarpeesta kohteille.

Kunnossapidon erikoistarpeista lähtien pyritään mallintamaan sellainen sumeaa logiikkaa hyödyntävä osasovellus jolla voidaan hallita mahdollisimman suuri joukko käytännön ongelmista. Jotta tähän on mahdollista päästä, niin sähkönjakeluyhtiöiden nykyiset kunnossapidon kokemuspohjaiset peukalosäännöt on määriteltävä formaalimpaan muotoon sumeiksi säännöiksi. Varsinaisen toteutuksen osalta tärkeimmät piirteet ovat yleiskäyttöisyys ja laajennettavuus.

Tässä työssä on myös sovellettu sumeaa logiikkaa alalle jossa sitä ei aiemmin ole käytetty. Paikkatiedonhallinta on tarkoista koordinaateista ja loogisista yhteyksistä huolimatta erittäin hyvin sumeaan päättelyyn sopiva koska siihen liittyy runsaasti sellaisia reaalimaailman käsitteitä, esimerkiksi lähellä, alueella tai samassa solmupisteessä verkossa, joille ei ole aiemmin ollut kunnollista ilmaisukeinoa tietojärjestelmissä. Myös nämä säännöt on ollut pakko kehittää itse koska formaalia, tietojärjestelmässä soveltamiseen kelpaavaa esitystä ei ole ollut olemassa.

Tämä työ jakautuu sisältönsä puolesta kolmeen osaan.
1. Lähtötilanteen ja teoreettisen pohjan esittely (luvut 2-4)
2. Sähkönjakeluverkoston kunnossapidon sumea säännöstö ja toteutus (luvut 5 ja 6)
3. Paikkatiedonhallinnan sumeat säännöt ja niiden toteutus (luvut 7 ja 8)

2 Käsitteet ja lähtötilanne

2.1 Paikkatiedonhallinta

Paikkatiedonhallinta on laaja käsite. Tässä työssä rajoitutaan käsittelemään paikkatietoa pelkästään kohteiden sijaintia kuvaavana ominaisuutena, jonka perusteella voidaan toimia ja tehdä päätelmiä. Termiin liittyvät muut käsitteet ja periaatteet sivuutetaan työhön liittymättöminä.

Paikkatiedonhallinta pohjautuu kohteille annettaviin sijaintitietoihin, jotka yleensä ovat jonkinlaisessa koordinaattimuodossa. Kohteet voidaan luokitella kolmeen pääluokkaan:

- Pisteet
- Viivat
- Alueet (alueiden rajat koostuvat viivoista)

Lisäksi kohteilla voi olla hierarkioita, jolloin alikohteella ei välttämättä ole omaa sijaintitietoa lainkaan. Kaikilla paikkatiedonhallintaan kuuluvilla kohteilla on kuitenkin joko omat sijaintitiedot tai välillisesti jostain toisesta kohteesta saatavat sijaintitiedot. Hierarkiat jakautuvat seuraaviin päätyyppeihin paikkatiedonhallinnan osalta:

- Alikohteella ei ole omia sijaintitietoja vaan sillä on sama sijainti kuin pääkohteella.
- Alikohteella on oma sijainti, mutta se ilmoitetaan suhteessa pääkohteeseen, ei normaalikoordinaatteina. Yleensä siirtymä on pieni.
- Alikohteella on oma sijainti; alisteisuudesta huolimatta se voi sijaita missä vain.

2.2 Verkkotietojärjestelmä

Tämän työn kannalta verkkotietojärjestelmällä käsitetään tietojärjestelmää jolla mallinnetaan jotakin reaalimaailman verkostoa s.e.verkoston kohteista saadaan paitsi loogiset yhteydet ja solmukohdat, myös tarkat sijaintitiedot kullekin verkon kohteelle. Esimerkkejä tällaisista verkostoista ovat sähkö-, puhelin- ja kaukolämpöverkot.

Verkkotietojärjestelmissä voidaan reittien sijaintitietoa käsitellä kahdella tavalla:

- Yksiviivaesityksenä jolloin samaa reittiä kulkevat kohteet (kaapelit, putket tms.) viittaavat viiva-alkioihin joilla ei tällöin ole muuta sisältöä kuin sijainti.
- Moniviivaesityksenä jolloin samaa reittiä kulkevat kohteet kuvataan vierekkäin kulkevina viivoina.

Solmukohdissa voidaan käyttää erityisiä pistemäisiä solmukohteita tai jättää niiden käsittely pelkän sijaintitiedon varaan. Usein sijaintikohteilla on hierarkista tietoa jolla ei ole omaa sijaintitietoa.

2.3 Nykyinen ohjelmisto

Tässä työssä käsiteltävä ohjelmisto on sähkönjakelulaitokssa käytössä oleva verkkotietojärjestelmä Xpower. Sen avulla sähkölaitokset hoitavat graafisesti mm. verkkojen dokumentoinnin, suunnittelun, kunnossapidon ja kytkentätilanteiden

reaaliaikainen hallinnan. Xpower sisältää useita toiminnallisia moduleita sähkönjakelulaitosten eri tarpeisiin:

- Verkon suunnittelu ja dokumentointi graafisesti
- Toiminnallisten karttojen ja kaavioiden tuottaminen
- Verkostolaskenta
- Käytön tuen toiminnot
- Kunnossapito
- Rakentamisen tietojärjestelmä
- Raporttien tuottaminen

Xpower käyttää normaalia relaatiotietokantaa kaikkeen tiedon tallentamiseen. Tarvittava osa verkostosta ladataan muistiin käsittelyn ajaksi.

2.3.1 Kunnossapitomoduli

Xpower-järjestelmän osana on kunnossapitomoduli, jolla voidaan hoitaa kunnossapidon suunnittelu ja toteutus. Sen toiminta on pitkälti käyttäjän määriteltävissä, koska tietokannassa voidaan määritellä kohteille ja kohdejoukoille tehtävät kunnossapitotoimenpiteet hyvin vapaasti. Käyttäjä voi lisäksi rajata käsiteltävät kohteet laitetyypin sisällä esim. alueen tai mittaustulosten perusteella.

Verkon kohteista on laajentuvan mittaus- ja tarkastustoiminnan seurauksena saatavissa kunnossapitotoimintaa tukevaa tietoutta kasvavassa määrin. Tietojen suuret määrät vaativat kuitenkin riittävän tehokkaan analysointivälineen. Tässä työssä on tarkoitus selvittää esimerkin avulla sumean logiikan tarjoamia mahdollisuuksia verkon kohteiden kunnossapidon suunnittelussa. Nykyisen modulin rakenne käsitellään tarpeellisella tarkkuudella.

3 Sähkönjakeluverkon kunnossapito

3.1 Johdanto

Suurin osa maamme sähkönjakeluverkosta on rakennettu viisi- ja kuusikymmentä-luvuilla, joten verkon yksittäiset komponentit jo ikänsä puolesta edellyttävät järjestelmällistä kunnonseurantaa. Uuden sähkömarkkinalain seurauksena syntyviltä verkkoyhtiöiltä edellytetään sähkön laadun ja verkon käyttövarmuustekijöiden hallintaa. Sen eräänä keskeisenä tekijänä on luotettavan sähkönjakelun varmistaminen. Jotta sähkön hinnan nousua voidaan rajoittaa, on jakeluverkkoa hoidettava mahdollisimman taloudellisella tavalla. Nykyisin kunnossapito perustuu pitkälti yksittäisiin havaintoihin verkon komponenttien tilasta sekä vika- ja häiriötilanteista saatuun informaatioon. Koska verkkoyhtiöt käyttävät kunnossapitoon vuosittain satoja miljoonia markkoja, jo pienikin parannus kunnossapidon hallinnassa ja suunnittelussa merkitsee tuntuvia säästöjä.[1]

3.2 Käsiteltävät komponentit

Tässä työssä käsitellään sähkönjakeluverkon komponenteista esimerkkinä tehomuuntajaa. Siihen liittyvät kunnossapidon käsittelysäännöt ovat helposti yleistettävissä muille verkon laitteille. Olennaisia muita komponentteja ovat erityyppisten muuntajien ohella katkaisijat, erottimet ja pylväät.

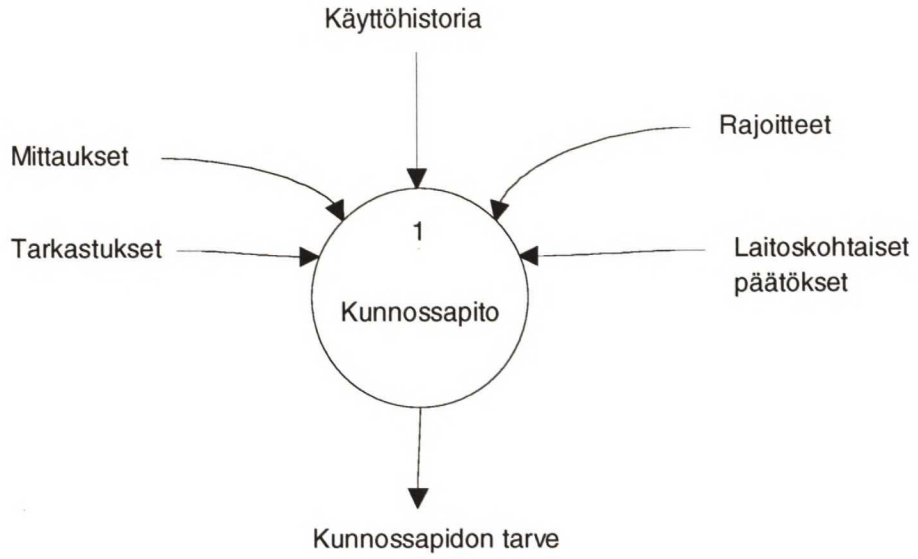
Tehomuuntaja on sähkönjakeluverkon komponentti, joka välittää sähköä eteenpäin verkossa muuttaen sen jotain ominaisuutta, yleensä tehoa. Tehomuuntaja voi myös haaroittaa sähkön useammalle johdolle.

Tehomuuntajat ovat kalliita laitteita, joten huolellisen kunnossapitotoiminnan seurauksena säästöt voivat olla merkittäviä. Tehomuuntajan kuntoa voidaan seurata usealla eri tasolla, esim. erilaisilla mittauksilla. Mittaustuloksien yhteisvaikutus ratkaisee muuntajan komponenttien vaikutuksen muuntajan kuntoon. Muuntajan mitattavia komponentteja/ominaisuuksia ovat mm.[1]

- Käämikytkin
- Käämikytkimen ohjain
- Läpiviennit
- Öljyt
- Paperieriste

Muuntajan kuluminen riippuu voimakkaasti käyttöprofiilista. Mikäli se joutuu toimimaan usein kapasiteettinsa äärirajoilla, sen odotettavissa oleva elinikä laskee. Ylikuormitustilanteet ovat yleensä lyhyitä, mutta ne lyhentävät huomattavasti muuntajan oletettua elinikää.[2]

3.3 Kunnossapitoon vaikuttavat tekijät

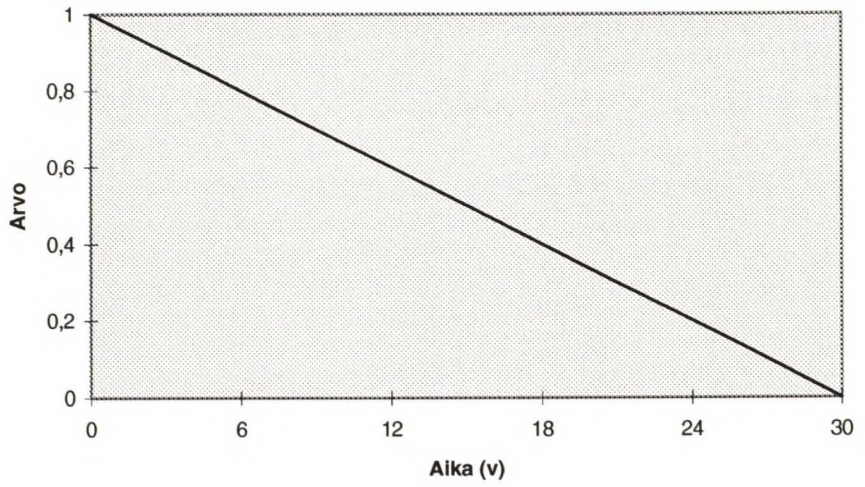


Kunnossapitoon vaikuttaa lähtötietoina mittausten ja tarkastusten tulokset sekä laskennalliset arviot muuntajan kunnosta. Käyttöhistoria käsittää aikariippuvan kulumisen lisäksi mm. tehdyt kunnossapitotoimenpiteet, ylikuormitustilanteet ja laitteen vikahistorian. Kaikkien tekijöiden yhteisvaikutus ratkaisee muuntajan kunnossapidon tarpeen.[1]

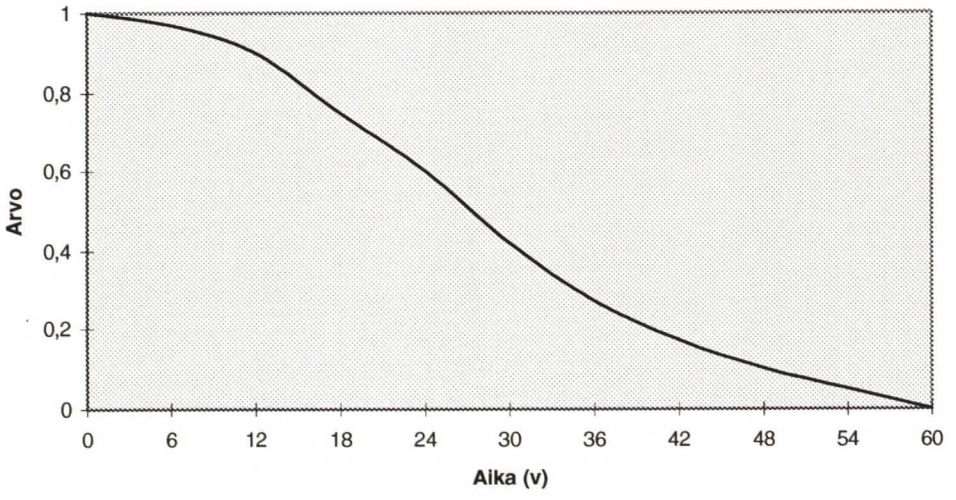
3.3.1 Aika

Kullekin muuntajalle on tiedossa valmistajan antama laskennallinen elinikä. Sitä pidempään laitetta ei yleensä kannata käyttää vaikka se vielä muuten olisikin riittävän hyvässä kunnossa mittaustietojen perusteella. Ajan perusteella tapahtuva kuluminen kuvataan kulumiskäyrällä. Kulumiskäyrästä saadaan muuntajan iän perusteella sen kuntoarvio. Mikäli mittaustietoja ei ole käytettävissä, voidaan päätelmiä tehdä myös pelkän kuntokäyrän perusteella. Tällä tavalla tehdyt johtopäätökset ovat kuitenkin luotettavuudeltaan heikkoja.

Kunnossapitotoimenpiteet vaikuttavat kulumiskäyrään ja muuntajan tämänhetkisen kuntoarvioon. Kaikilla muuntajilla pidetään kuitenkin maksimi-ikä jonka jälkeen ne pyritään viimeistään korvaamaan uudella. Jos mitattu kunto on huono, vaihtoa voidaan aikaistaa ja vastaavasti jos havaittu kunto on hyvä, sitä voidaan viivästyttää.



Kuva 3: Esimerkki kulumiskäyrästä: Lineaarinen kulumiskäyrä



Kuva 4: Esimerkki kulumiskäyrästä: ensin kiihtyvä, sitten hidastuva kuluminen

3.3.2 Mittaukset

Muuntajien kunnosta saadaan huomattavasti luotettavampi arvio tekemällä säännöllisesti mittauksia niiden eri ominaisuuksista. Mittaustuloksia käsitellään kahdella tavalla:

- Selkeä lukuarvo jota voidaan suoraan verrata sallittuihin raja-arvoihin; esimerkiksi vastus.
- Mittaajan tekemä luokittelu ilman minkäänlaista lukuarvoa; esimerkiksi korroosioarvio.

Mittaukset tarkentavat tietoa muuntajan kunnosta. Niissäkin on virhemarginaali, mutta olennaisesti pienempi kuin pelkällä elinikälaskennalla saatu. Yksittäinen mittaustulos ei vaikuta kovin paljoa ellei se alita määriteltyä kriittistä raja-arvoa. Luokituksissa on samoin määriteltävä niiden hyväksyttävyys. Aina kun mittaustulos saavuttaa kriittisen raja-arvon, se johtaa kunnossapitotoimenpiteisiin.

Vanhentuneita mittaustuloksia ei huomioida lainkaan tai ne saavat alhaisemman painoarvon muuntajan kuntoa arvioitaessa.

3.3.3 Kunnossapidon toimenpiteet

Muuntajille tehtävät kunnossapitotoimenpiteet luokitellaan tässä dokumentissa 4 pääluokkaan käsittelysääntöjensä perusteella. Eri muuntajatyypeille määritellään niille ominaiset toimenpiteet, aikavälit yms. Kaikista muuntajalle tehdyistä toimenpiteistä talletetaan merkintä tietokantaan.

Tehdyt huolto- ja korjaustoimenpiteet vaikuttavat muuntajan kuntoarvioon. Ne vaikuttavat suoraan tyyppikohtaiseen kulumiskäyrään joka määrittelee laitteen kunnon ajan perusteella. Tyypillisesti toimenpiteet nostavat kuntoarvion paremmalle tasolle kulumiskäyrältä, josta kuluminen jatkuu alkuperäisellä nopeudella. Toimenpiteet vaikuttavat näin periaatteessa koko muuntajan käyttöajan. Eri toimenpiteet vaikuttavat kuitenkin eri tavoin.

3.3.3.1 Mittaus- ja tarkastushuolto

Muuntajan kunnosta mitataan laitetyyppikohtaiset suureet tai luokitellaan sen kunto annettuihin luokkiin. Mittaus- ja tarkastushuollot käsitellään tässä työssä samoilla periaatteilla. Jatkossa puhutaan pelkästä mittaushuollosta.

Mittaushuoltoja tehdään säännöllisin väliajoin koska pelkkä ajan perusteella arviointi ei takaa riittävää luotettavuutta. Mittaustuloksia käytetään aina kunnonarvioinnin pohjana. Mikäli tulokset ovat vanhentuneita, niille annetaan enemmän virhemarginaalia huonompaan suuntaan. Vain viimeisimmällä mittaustuloksella on merkitystä muuntajan kuntoa arvioitaessa. Monille tuloksille pidetään kuitenkin yllä historiatietoja joista nähdään arvojen kehitys pidemmällä aikavälillä.

3.3.3.2 Määräaikaishuolto

Säännöllisin aikavälein tehtävä huolto jossa muuntaja puretaan ja huolletaan perusteellisesti, kaikkia osakomponentteja myöten. Kaikki komponentit tarkastetaan ja tarpeen vaatiessa vaihdetaan. Valmistajien ilmoittamat eliniät olettavat yleensä säännölliset määräaikaishuollot. Määräaikaishuoltojen ajoituksessa on jonkin verran joustovaraa; sen vuoksi niitä ei lasketa mukaan kulumiskäyriin. Toimenpiteellä saadaan ylläpidettyä muuntajien kuntoa niin että ne säilyvät käyttökelpoisina elinikänsä loppuun saakka.

3.3.3.3 Korjaus

Muuntajassa havaittu tai oletettu vika korjataan. Osakomponentteja saatetaan vaihtaa korjauksen yhteydessä mutta ne käsitellään erillisinä toimenpiteinä. Muuntajan kunto paranee korjauksen seurauksena, mutta siitä ei tule uuden veroista. Muuntajalle ei tehdä mitään korjaukseen liittymättömiä huoltotoimenpiteitä.

3.3.3.4 Vaihto

Muuntaja ja kaikki sen osat korvataan uudella vastaavalla. Mikäli osa komponenteista säilytetään, niiden uutta heikompi kunto pitää muuntajan kokonaiskunnon heikompana.

3.3.4 Historia

Muuntajille talletetaan huoltojen lisäksi myös muuta tietoa niiden käyttöhistoriasta. Olennaisia talletettavia tietoja ovat

- Laitteen käyttöprofiili (mahdolliset ylikuormitukset yms.)
- Aiempi vikaantuminen ja vikojen vakavuusluokitukset (riippuen mm. katkeaako sähkön kulku). Vioista talletetaan myös varsinaiset syyt jos ne ovat tiedossa, jotta voidaan tehdä päätelmiä mahdollisista alue- tai laitetyyppikohtaisista suuremmista vikaantumistodennäköisyyksistä.

Historiatiedot ovat eräänlainen mittauksien ja toimenpiteiden välimuoto: ne kuvaavat menneisyyttä kuten tehdyt toimenpiteetkin, mutta niistä tehdään päätelmiä muuntajan kunnosta kuten mittaustiedoista. Historiatietoihin voidaan soveltaa joko tilastollista analyysiä tai käyttää suoraa laskennallista vaikutusta. Tilastollista analyysiä sovelletaan vikatietoihin ja mittaustuloksiin.

Suoraa laskennallista vaikutusta voidaan soveltaa esimerkiksi ylikuormituksen vaikutukseen muuntajan elinikään. Laskennallinen vaikutus ei ulotu kuin yhteen arvoon kerrallaan. Käsittelytapa on analoginen huoltojen kanssa jotka muuttavat kulumiskäyrältä saatua vanhenemisarvoa.

3.3.5 Muuntajan ominaisuudet

Kullakin muuntajatyypillä on omat ominaisuustietonsa. Kunnossapitojärjestelmää varten on lisäksi määriteltävä kullekin laitetyypille

1. Laitetyypille tehtävät mittaukset
2. Mittausten sallitut arvot (raja-arvoina tms.)
3. Mittausten luokitukset (jollei lukuarvoja voida antaa)
4. Laitetyypin osakomponenttien hierarkia ja osien saama painoarvo laitteen kokonaiskuntoarviossa.

5. Vastaavat tiedot kaikille osakomponenteille (hierarkian laitteiden kuntoarviot on laskettava ennen käsiteltävää kohdetta)
6. Kunnossapitotyyppi joka määrittelee muuntajalle ominaiset operaatiot. Laitetyyppi kuuluu kokonaisuudessaan johonkin kunnossapitotyyppiin. Määriteltäviä ominaisuuksia ovat mm. määräaikaishuoltoväli.

3.4 Tulokset ja käyttö

Kunnossapidon järjestelmä tuottaa arvion kuinka suuri jonkin tietyn muuntajan huollon tarve on. Tällainen arvio saadaan aikaan esim. antamalla muuntajille kuntopisteitä mittaustuloksien perusteella ja pistemäärälle minimiarvo. Tavoitteena on kuitenkin saada optimoitua automaattisesti kunnossapitotoimintaa s.e. kokonaiskustannukset saadaan minimoitua. Kokonaiskustannuksissa otetaan huomioon paitsi suorat huoltokustannukset, myös keskeytykset, vikojen hinta, pitkän aikavälin kustannukset yms. Huomioitavaa on myös erilaiset rajoitukset ja laitoskohtaiset painotukset kunnossapidossa. Seuraavissa kappaleissa esitetään optimoinnin periaatteet ja siinä huomioon otettavat tekijät.

3.4.1 Kunnossapidon optimointi

Sähkönjakeluverkoston kunnossapitoa suunnitellaan useita vuosia eteenpäin. Suunnitelmat ovat sitä tarkempia mitä lähempänä toteutusajankohta on. Lähiaikoina tehtävät huollot pyritään jaksottamaan niin, että ne aiheuttavat mahdollisimman vähän haittoja ja kustannuksia. Olennainen vaikutus on sillä vaatiiko kyseinen toimenpide keskeytyksen sähkön kulkuun.

Useimmille kunnossapitotehtäville on joku kokemusten perusteella laskettu aika jolloin työ tulisi tehdä. Toimenpiteiden ajoitus ei kuitenkaan ole mitenkään kiinteä vaan niitä voidaan siirtää aiemmaksi tai myöhemmäksi, esimerkiksi työvoiman tai koneiden riittävyyden perusteella. Siirtämisessä, erityisesti myöhentämisessä, on olennaista riskien hallinta. Kunnossapito on aina harkittujen riskien ottamista. Viat ja katkokset ovat realisoituneita riskejä.

Joitakin huoltoja ei saa siirtää lainkaan. Kaikille huolloille määritellään järjestelmään siirrettävyysarvo joka on painokerroin sille mitä huoltoja aletaan ensimmäisinä siirtämään muuhun aikajaksoon.

Kunnossapidossa tehdään myös jonkun verran painotuksia kohteen keskeisyyden ja asiakkaiden merkityksen perusteella; yksittäisille kohteille tai kohderyhmille voidaan antaa erityisiä painoarvoja. Esim. teollisuusalueelle menevää verkon osaa tarkastetaan useammin kuin omakotitaloalueen jakeluverkkoa.

Yksittäisiä kunnossapito-operaatioita pyritään optimoimaan niin, että niiden välillä oleva joutoaika kuten matkat ja varusteiden kasaaminen jäävät mahdollisimman vähäisiksi. Mikäli toimenpiteitä tehdään samassa paikassa lyhyellä aikavälillä, operaatiot yhdistetään ja tehdään samalla kertaa. Toimenpiteitä yhdistellään siis pikemminkin paikan kuin toimenpiteiden samankaltaisuuden mukaan. Tarvikkeet ja operaatioiden kesto rajoittavat tämänkaltaista optimointia.

Kokonaiskustannuksien optimoinnissa tärkein sääntö on se että kannattaako muuntajaa korjata tai huoltaa lainkaan. Uusi laite voi tulla edullisemmaksi kuin vanhan korjaaminen. Samoin mikäli muuntajan laskennallinen elinikä on jo hyvin vähissä, voidaan turhan korjauksen välttämiseksi sen vaihtoa aikaistaa. Jos muuntajalle on jouduttu tekemään paljon korjauksia ja se vikaantuu huomattavasti useammin kuin pitäisi, on myöskin perusteltua harkita sen korvaamista uudella yksilöllä tai laitetyypillä, varsinkin jos sen

kokonaiskustannukset odotettavissa olevana elinikänä ovat olennaisesti suuremmat kuin sen korvaamiskustannukset.

Kunnossapitotehtävien ajoituksen optimointia tehdään monella aikavälillä: vuorokaudenaika, viikonpäivä, vuodenaika. Katkokset pyritään ajoittamaan yöaikaan jolloin kotitalouksien aiheuttama kuormitus on pienempää; teollisuuslaitoksilla tämän jaksotuksen hyöty on pienempi. Töitä ei yleensä tehdä viikonloppuna vaikka asiakkaille koituvat haitat jäisivätkin pienemmiksi, koska työkustannukset ovat suuremmat. Suurin osa kunnossapitotöistä tehdään kesäaikaan, koska silloin töiden teko on helpompaa. Talvella lumi ja routa aiheuttavat ylimääräisiä hankaluuksia. Lisäksi kesällä myös kuormitus on pienempi lämmitys- ja valaistustarpeiden vähäisyyden vuoksi. Mittaus- ja tarkastustehtäviä joudutaan tekemään ympäri vuoden.

Kustannuksia pyritään tasaamaan vuositasolla. Tämä on välttämätöntä jonkinlaisen ennustettavuuden saamiseksi ja budjetoinnin helpottamiseksi. Jos jonakin vuonna on suuri joukko isoja huoltoja, on osa niistä mahdollisesti siirrettävä edelliselle tai seuraavalle vuodelle. Tämänkaltainen optimointi vaatii hintatietoja joko operaatiokohtaisesti tai komponenttien ja työvoiman kustannuksia.

3.4.2 Rajoitteet

Kunnossapitotoimenpiteiden ajoittamisessa on optimoinnin ohella otettava huomioon eräitä absoluuttisia rajoituksia. [1] Nämä ovat käyttäjän määrittelemiä arvoja mutta töiden järjestelyn kannalta muuttumattomia. Ensinnäkin riskinhallinta vaatii että järjestelmässä on oltava jokin yläraja kuinka paljon jotakin huoltotoimenpidettä voidaan siirtää ilman että käyttäjältä kysytään varmistusta. Tämä pätee ennenkaikkea huoltojen myöhentämiseen, mutta myös aikaistamiseen koska tällöin väliaika seuraavaan huoltoon kasvaa. Raja määritellään prosentuaalisena arvona.

Toinen ehdoton raja kuinka paljon operaatioita voidaan tietyllä aikavälillä tehdä on resurssien (esim. työvoima) riittävyys. Tämän laskenta edellyttää että toimenpiteille on olemassa arvio kuinka paljon henkilökuntaa, koneita ja muita resursseja sekä aikaa tehtävä vaatii.

Kunnossapitoon on varattu tietty määrä rahaa yrityksen/laitoksen budjetissa. Summa määräytyy yleensä kunnossapidon pitkän aikavälin kustannuksien keskiarvosta joten sillä pitäisi pystyä hoitamaan tarpeelliset tehtävät. Optimoinnissa mainittu kustannusten tasaaminen on yksi osa tätä rajoitetta. Mikäli kustannukset tasattunakin ylittävät budjetin tälle vuodelle, on tutkittava voidaanko siirtää seuraaville vuosille joitakin isompia töitä. Mikäli tällaista mahdollisuutta ei ole, käyttäjää on informoitava rahan riittämättömyydestä. Budjetti vaikuttaa järjestelmään ainoastaan vuosikohtaisen kustannusmaksimin kautta.

Kustannuslaskenta edellyttää tietoja komponettien hinnoista ja työvoima-kustannuksista. Kustannukset voidaan laskea myös suoraan toimenpidekohtaisista kustannusarvioista. Ilman hintatietoja kustannuksien tarkkailu on mahdotonta.

3.4.3 Huoltojen aikataulu

Järjestelmä tuottaa tietylle aikavälille listan töistä jotka sinä aikana on tehtävä. Listat voivat olla myös aluekohtaisia, kunnossapito hoidetaan ainakin isommissa yhtiöissä piireittäin. Työlista ei ylitä aikavälillä käytettävissä olevaa työvoiman määrää eikä vuoden maksimikustannuksia, tai siitä ilmoitetaan käyttäjälle.

Äkilliset vakavat viat voivat myös aiheuttaa muutoksia työlistoihin. Yleensä tällaisia töitä varten on jätetty hieman henkilökuntaa reserviin, joten ne eivät vaikuta kuin ääritapauksissa tai useiden vikojen tapauksessa (esim. myrskyn jälkeen). Listat on oltava helposti tuotettavissa uudestaan tällaisten tapausten varalta.[1]

3.4.4 Elinikä

Kaikilla laitteilla on valmistajan ilmoittama oletettu elinikä. Tätä käytetään yleensä ylärajana sille kuinka kauan muuntajaa kannattaa käyttää; laiteyksilöiden välillä voi kuitenkin olla huomattavia eroja. Elinikä on laskennallinen suure josta voidaan tehdä päätelmiä muuntajan kunnosta silloin kuin tarkempaa tietoa ei ole saatavilla. Huolto- ja korjaustoimenpiteet lisäävät laitteiden elinikää.

Elinikäarvio voi muuttua dynaamisesti ajan kuluessa. Mikäli laitetyypin havaitaan olevan erityisen herkkä joillekin ympäristöolosuhteille, sellaisessa ympäristössä olevien muuntajien elinikäennustetta alennetaan valmistajan ilmoittamasta. Jos jokin muuntaja vikaantuu jatkuvasti, sen elinikää voidaan arvioida uudelleen. Toisaalta, mikäli laitteen mittaustulokset ovat johdonmukaisesti paremmat kuin mitä laskennallinen elinikä antaisi odottaa, ennustetta voidaan myös parantaa. Kaikki eliniän perusteella tehtävät päätelmät pohjautuvat kullakin hetkellä voimassa olevaan arvioon.

Muuntajien huoltokustannuksilla ja vikaantumistiheyksillä on taipumusta kasvaa niiden vanhetessa. Usein laitteiden taloudellinen elinikä jää lyhyemmäksi kuin niiden laskennallinen elinikä. Tämä otetaan huomioon laitteiden eliniän määrityksessä ja mikäli huoltokustannukset osoittautuvat suuremmiksi kuin arvioitu, on elinikäarviota alennettava vastaamaan taloudellista elinikää.

Kohteen iän saavuttaessa elinikäarvion on useimmiten perusteltua vaihtaa se uuteen vastaavaan. Mikäli käytännön kokemukset (esim. mittaustulokset) osoittavat elinikäarvion virheelliseksi, voidaan laitteen vaihtoa viivyttää pienellä riskillä. Tällaista muuntajaa kuitenkin valvotaan tarkemmin kuin normaalisti. Vastaavasti normaalia huonompi muuntaja voidaan vaihtaa jo aiemmin.

3.4.5 Laiteryhmät

Kunnossapitojärjestelmässä voidaan tehdä tilastollisia analyysejä paitsi laitetyypeittäin, myös sijainnin, priorisoinnin, mittausten (laitetyypit joille tehdään jokin tietty mittaus) ja kunnossapitotyyppien mukaan. Eri luokille ja niiden yhdistelmille voidaan tehdä päätelmiä kunnossapidon tarpeista ja suunnitelmia laiteryhmien tulevasta kunnossapidosta tällä perusteella. Esimerkiksi voidaan havaita että tietyn alueen muuntajat kaipaavat erityistä tarkkailua lintujen suuren määrän vuoksi. Vertailu tehdään usein kaikkien laitteiden keskiarvoon tai vastaavaan tilastolliseen arvoon. Laiteryhmien käytöksen tutkiminen vaatii usein historiatietoja pidemmältä ajalta, esimerkiksi vikaantumistiedot.

Tilastollisella analyysillä haetaan myös ryhmälle tavallista tyypillisempiä vikoja ja vertaillaan ryhmien vikaantumistiheyttä. Laitetyyppien tapauksessa tämä merkitsee sitä että voidaan karsia vikaantumisalttiit laitetyypit pois hankinnasta tai käytöstä. Erityisesti jos laitetyypillä on jokin tyypillinen vika, se on havaittavissa tällaisella tilastollisella analyysillä.

Mikäli vika ilmenee vain tietyissä olosuhteissa, vian yleisyyden havaitseminen voi olla vaikeampaa. Esimerkiksi laitetyypin kosteusarkuus on havaittavissa mikäli yhdistetään laitetyyppi- ja ympäristöryhmittely. Lämpötilariippuvuus on myöskin vaikeasti havaittavissa, koska se riippuu käytännössä vuodenajasta. Mikäli kohteet eivät muuten vikaannu lainkaan tai erittäin vähän, tällainen käytös näkyy myös vuotuisessa vikaantumisessa. Muutoin ainoa keino havaita tällaista käytöstä on lyhentää tarkasteluväli neljännesvuoteen tai vielä lyhyemmäksi. Samalla voidaan havaita muitakin aikariippuvia vikaantumispiikkejä.

Mikäli vikoja havaitaan kaikilla laitetyypeillä tietyllä alueella, voidaan päätellä alueen ympäristössä olevan jotakin laitteille haitallista. Esimerkkejä tällaisesta ovat ilman suuri saastepitoisuus, hyönteisten tai lintujen suuri määrä ja ilkivalta. Aluekohtaisessa tarkastelussa tällaiset vaikutukset tulevat helposti esille kaikilla aikaväleillä.

4 Teoreettinen pohja

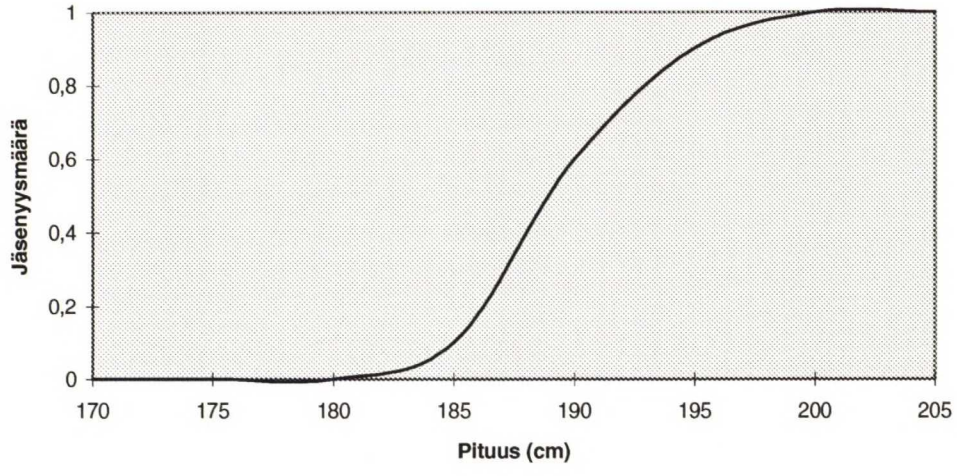
4.1 Sumeat joukot

4.1.1 Sumeiden joukkojen merkitys

Normaaleilla ei-sumeilla joukoilla (crisp set) alkio joko on tietyn joukon jäsen tai se ei ole. Sumeilla joukoilla (fuzzy set) alkiolla voi olla erilaisia jäsenyyden asteita: se voi olla kokonaan jäsen, se voi olla olematta ollenkaan jäsen, tai se voi olla jonkin verran jäsen. Sumeat joukot ovat eräänlaisia funktioita, jotka määrittävät alkioille niiden jäsenyysasteen, välillä nollasta yhteen (f:D→[0,1]). Nolla merkitsee, että alkio ei ole lainkaan joukon jäsen ja yksi että se on täysin joukon jäsen. Välillä on täysin jatkuva arvo, joka kuvaa jäsenyyden astetta, kuinka hyvin arvo vastaa joukkoa.[4]

Sumeiden joukkojen idea soveltuu erityisesti kielellisten suureiden kuvaamiseen. Kielen käsitteet ovat usein sellaisia ettei niitä voi ilmaista millään tarkalla matemaattisella arvolla. Silti ne ovat hyvin tavanomaisia arkikielessä. Esim. käsitteet 'pitkä' ja 'painava' ovat epämääräisiä eikä niille ole löydettävissä mitään tarkkaa raja-arvoa jonka jälkeen ne olisivat tosia ja jonka alapuolella epätosia. Niitä voidaan kuitenkin kuvata s.e. määritellään joukko pituuksia ja niitä vastaavat jäsenyyden määrät joukossa 'pitkä'. Tästä saadaan jatkuva käyrä annettujen ääripäiden välille. Tällöin voidaan määritellä mille tahansa annetulle pituudelle missä määrin se kuuluu joukkoon 'pitkä'.

Esim. Ihmisen pituudelle voidaan määritellä että alin pituus joka kuuluu jossain määrin joukkoon pitkä on 180 cm ja 200 cm on jo täysin pitkä. Esimerkiksi 175 cm pituinen henkilö ei ole pitkä; henkilö joka on 202 cm pituinen on selvästikin pitkä. Henkilö joka on 193 cm pituinen vastaa joukkoa 'pitkä' asteella 0,85.

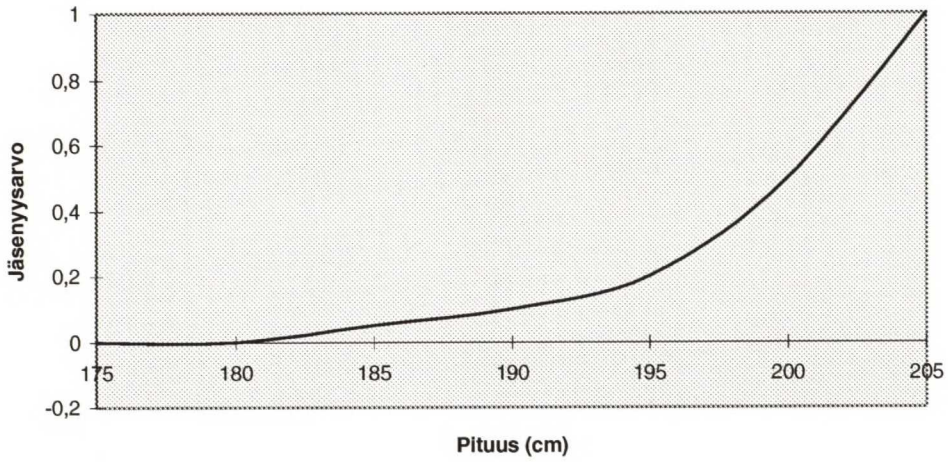


Kuva 1: Pitkä - sumean joukon jäsenyysfunktio

4.1.2 Suodattimet

Normaalissa kielessä on usein myös tapana korostaa jotain termiä lisämääreillä kuten 'hyvin', 'erittäin' tai 'melko'. Sumeassa logiikassa tällaisia kutsutaan suodattimiksi (hedge); ne muuttavat alkuperäisen jäsenyysfunktion muotoa ja/tai rajoja. Suodattimet ovat eräänlaisia funktioita jotka muuttavat sumeita arvoja (f:[0,1]→[0,1]).

Esimerkiksi 'hyvin pitkä' voi tarkoittaa sitä että alle 185-senttiset eivät kuulu joukkoon lainkaan ja funktio nousee jyrkemmin loppuosassaan. Melko-suodattimella olisi juuri päinvastainen vaikutus.



Kuva 2: Hyvin pitkä - sumean joukon jäsenyysfunktio

4.2 Sumea logiikka

4.2.1 Sumea ja Boolen logiikka

Normaali ei-sumea logiikka pohjautuu täydelliseen poissulkevuuteen. Kukin alkio joko on joukon jäsen tai ei ole. Myös joukkojen keskinäisissä suhteissa toimitaan tämän periaatteen pohjalta. Joukkojen yhdistämisessä on kaksi perusperiaatetta:[5]

1. Joukon ja sen komplementtijoukon leikkaus on tyhjä joukko: $S_1 \cap \sim S_1 = \varnothing$
2. Joukon ja sen komplementtijoukon unioni on koko arvoalue: $S_1 \cup \sim S_1 = X$

Normaalit Boolen logiikan perusoperaatiot toimivat näihin nojautuen:

- AND: vastaa joukkojen leikkausta
- OR: vastaa joukkojen unionia
- NOT: vastaa joukon komplementtia.

A and ~A = FALSE
A or ~A = TRUE

Säännöt kuvautuvat tulosjoukkoon, jota voidaan käyttää vastaavasti lähtötietona seuraavalle säännölle. Tällä tavalla pienemmistä säännöistä saadaa koostettua mikä

tahansa isompi sääntö. Säännön laskennan tuottaessa nollasta poikkeavan lopputuloksen, sanotaan että sumea sääntö on lauennut. Laukeamisaste on säännön saama arvo.

Sumeassa logiikassa nämä periaatteet eivät päde. Sumean joukon komplementtijoukko saa nollasta poikkeavia arvoja kaikkialla missä varsinainen joukko saa yhdestä poikkeavia arvoja. Esimerkiksi jos A = 0,6 niin ~A voi olla 0,4. Tällöin joukon ja sen komplementin leikkaus on erisuuri kuin tyhjä joukko. Tarkka arvo riippuu laskentatavasta mutta olennaista on se että se eroaa tyhjästä. Samoin unioni voi tuottaa alle 1:n olevia arvoja joukkojen yhteisellä alueella.

4.2.2 Sumean logiikan käsittelysäännöt

Sumeassa logiikassa on samat kolme perusoperaatiota kuin Boolen logiikassakin: and, or ja not. Niiden määrittely eroaa kuitenkin Boolen logiikan vastaavista, ja kustakin on olemassa lukuisia erilaisia variaatioita. Variaatiot tuottavat hieman erilaisia lopputuloksia samoilla lähtötiedoilla. Myöskään lausekkeiden sieventäminen ei tapahdu samalla tavalla kuin tavanomaisessa logiikassa, johtuen ym. keskinäisen poissulkevuuden periaatteen pätemättömyydestä. Esimerkiksi De Morganin sääntö (not(A and B) = not A or not B) ei päde sumeassa logiikassa.[4]

Sumean logiikan operaatioille on tavanomaista että tulos pienenee koko ajan mutta ei voi saavuttaa nollaa. Yleensä tämän vuoksi on määritelty joko yleinen tai sääntökohtainen raja-arvo (alfa-arvo tai nollatoleranssi), jota pienemmät totuusarvot tulkitaan tarkalleen nollaksi. Tätä raja-arvoa tarkastellaan kaikille välituloksille ja lopputuloksille.

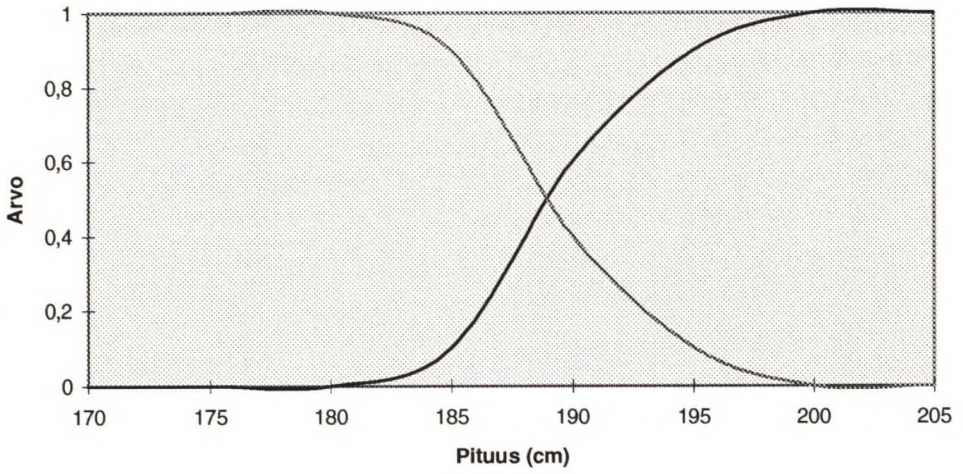
4.2.2.1 Not

Negaatio-operaatio not kohdistuu aina yhteen totuusarvoon. Boolen logiikassa TRUE muuttuu FALSE:ksi ja päinvastoin; sumean logiikan epätarkoilla arvoilla määrittely on liukuvampi. Kaikkien periaatteena on se että isosta totuusarvosta ($\mu_A [x] > 0,5$) tulee pieni ($\mu_A [x] < 0,5$) ja päinvastoin. Tässä dokumentissa kuvattavassa järjestelmässä tuetaan allaolevia negaatio-tyyppejä[1]:[4,6]

- Zadeh: $\sim\mu_A [x] = 1 - \mu_A [x]$
- Yager: $\sim\mu_A [x] = (1 - \mu_A [x]^k)^{1/k}$, $k \in]0, 5[$[2]
- Sugeno: $\sim\mu_A [x] = (1 - \mu_A [x]) / (1 - k * \mu_A [x])$
- Threshold (raja-arvo): $\sim\mu_A [x] = \begin{cases} 1, \text{ jos } \mu_A [x] < k \\ 0, \text{ jos } \mu_A [x] \geq k \end{cases}$
- Kosini: $\sim\mu_A [x] = 1/2*(1+\cos(\pi*\mu_A [x]))$

[1] Tähän työhön on valittu useita eri tyyppisiä not-, and- että or-operaatioita siten, että ne kattavat tärkeimmät eri tyypit. Pois jätetyt ovat joko tarpeettoman monimutkaisia tai pieniä variaatioita jo esiteltyihin.

[2] Arvolla k = 1 vastaa Zadeh:in not:ia.

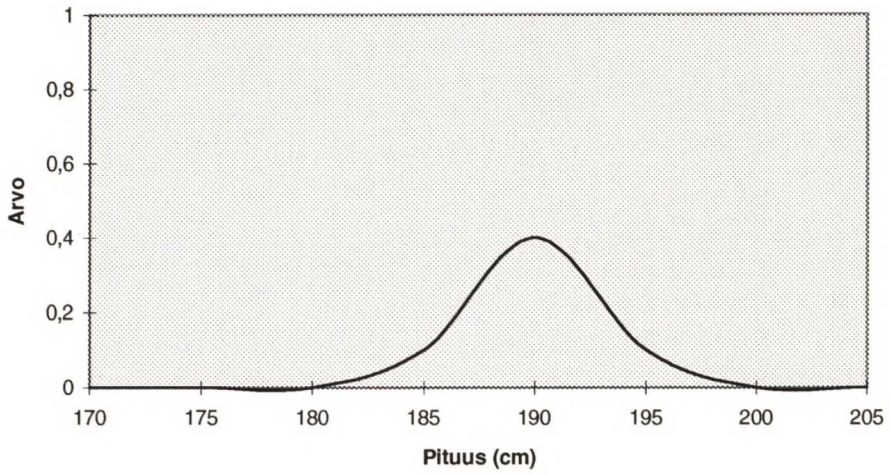


Kuva 3: Esim. Pitkä ja not pitkä, Zadeh not

4.2.2.2 And

Leikkaus-operaatio and kohdistuu aina kahteen totuusarvoon. Boolen logiikassa TRUE and TRUE = TRUE, kaikilla muilla lähtöarvoilla tulos on FALSE. Sumean logiikan and-operaatioiden periaatteena on se, että pienempi kahdesta totuusarvosta hallitsee lopputuloksessa ja suuremman vaikutus on vähäisempi tai olematon. Tässä dokumentissa kuvattavassa järjestelmässä tuetaan allaolevia and-tyyppejä[2]:[4,6]

- Zadeh: $\min(\mu_a [x], \mu_b [y])$
- Keskiarvo: $(\mu_a [x] + \mu_b [y]) / 2$
- Keskiarvo^2: $((\mu_a [x] + \mu_b [y]) / 2)^2$
- √Keskiarvo: $\sqrt{((\mu_a [x] + \mu_b [y]) / 2)}$
- Tulo: $(\mu_a [x] * \mu_b [y])$
- Rajattu summa: $\max (0, (\mu_a [x]+\mu_b [y]-1))$
- Yager: $1 - \min(1, ((1-\mu_a [x])^k + (1-\mu_b [y])^k)^{1/k})$

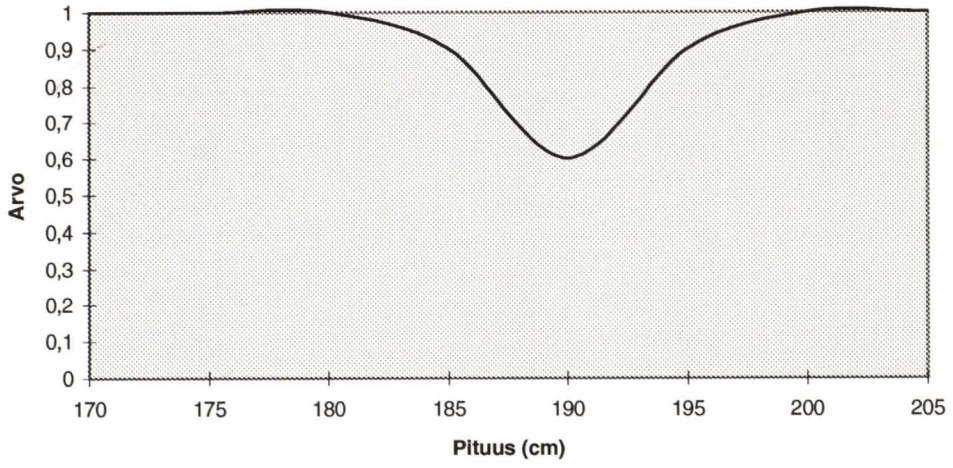


Kuva 3: Esim. Pitkä and not pitkä

4.2.2.3 Or

Yhdiste-operaatio or kohdistuu aina kahteen totuusarvoon. Boolen logiikassa FALSE or FALSE = FALSE, kaikilla muilla lähtöarvoilla tulos on TRUE. Sumean logiikan or-operaatioiden periaatteena on se, että suurempi kahdesta totuusarvosta hallitsee lopputuloksessa ja pienemmän vaikutus on vähäisempi tai olematon. Tässä dokumentissa kuvattavassa järjestelmässä tuetaan allaolevia or-tyyppejä[2]:[4,6]

- Zadeh: $\max(\mu_a [x], \mu_b [y])$
- Keskiarvo: $(2*\min(\mu_a [x], \mu_b [y]) + 4*\max(\mu_a [x], \mu_b [y])) / 6$
- Keskiarvo^2: $((2*\min(\mu_a [x], \mu_b [y]) + 4*\max(\mu_a [x], \mu_b [y])) / 6)^2$
- √Keskiarvo: $\sqrt{((2*\min(\mu_a [x], \mu_b [y]) + 4*\max(\mu_a [x], \mu_b [y])) / 6)}$
- Tulo: $(\mu_a [x] + \mu_b [y]) - (\mu_a [x] * \mu_b [y])$
- Rajattu summa: $\min(1, (\mu_a [x]+\mu_b [y]))$
- Yager: $\min(1, (\mu_a [x]^k + \mu_b [y]^k)^{1/k})$



Kuva 3: Esim. Pitkä or not pitkä

5 Kunnossapidon sumea järjestelmä

Kaikkien sääntöjen lopputuloksena on totuusarvo välillä [0,1]. Kaikki lähtöarvot (mittaustulokset yms.) muunnetaan tälle välille ennenkuin niitä käytetään. Vasta käyttäjälle näytettävät arvot luokitellaan reaalimaailman arvoihin tai käsitteisiin. Kuntoarvio 1 tarkoittaa ehjää, 0 käyttökelvotonta.

Kaikkien sääntöjen laskenta tapahtuu laskentahetkellä käytettävissä olevilla tiedoilla. Lopputulos muodostuu hetkellisestä näkymästä, ja lähtötietojen muuttuessa aiemmin laskettu lopputulos ei enää täysin vastaa todellisuutta.

Säännöissä on käytetty kolmenlaisia loogisia operaatioita: and, or ja not. Nämä tarkoittavat sumean logiikan vastineitaan, mutta ei ole mitenkään yleisesti määriteltävissä mitä nimenomaista versiota kustakin käytetään. Tämä määritellään sääntökohtaisesti parametritietona, jotta saadaan järjestelmän säätäminen mahdolliseksi kaikilta osin. Eri versioilla on joitakin toivottuja ja ei-toivottuja ominaisuuksia, esim. tuleeko and-operaation tulokseksi nolla jos toinen argumentti on nolla ja toinen ei.

Totuusarvoja käytetään ainoastaan rajallisella tarkkuudella, arvot pyöristetään 0,01:n tarkkuudella. Säännöille määritellään myös minimiarvo jonka alle menevät totuusarvot tulkitaan tarkasti nollaksi. [4] Osalla säännöistä tämä minimi voi olla 0, esimerkiksi termisessä kulumisessa minimien asettaminen ei ole mielekästä koska siitä saatavat arvot ovat aina vain pohjana reaalimaailman tuloksista saataville arvoille. Mittaustuloksissa hyväksyttävän minimin asettaminen on useimmiten tarpeen.

Kaikkiin kunnossapidon sääntöihin voidaan myöhemmin lisätä suodattimia joilla saadaan hienosäädettyä järjestelmän toimintaa. Eniten suodattimilla on kuitenkin käyttöä mittaustuloksien muokkaamisessa.

5.1 Termit

K_x	x:n arvo
N_x	x:n lukumäärä
C_x	x:n kustannukset
S_x	x:n arvo muunnettuna sumeaan skaalaan
t	aika
min	Pienempi kahdesta arvosta
MIN_x	Pienin joukosta arvoja (mahd. rajattuna x:n perusteella)
max	Suurempi kahdesta arvosta
MAX_x	Suurin joukosta arvoja (mahd. rajattuna x:n perusteella)
OR_x	Arvojoukon x tai-operaatio
AND_x	Arvojoukon x ja-operaatio

5.2 Vanheneminen (terminen kuluminen)

Määritellään N kappaletta aikavälejä, kuukauden tarkkuudella, ja niille kullekin kuntoarvo [0,1]. Aikojen väliin jäävät arvot voidaan laskea lineaarisella approksimaatiolla. Jako ja arvot ovat laitetyyppikohtaiset.

5.3 Huoltojen vaikutus

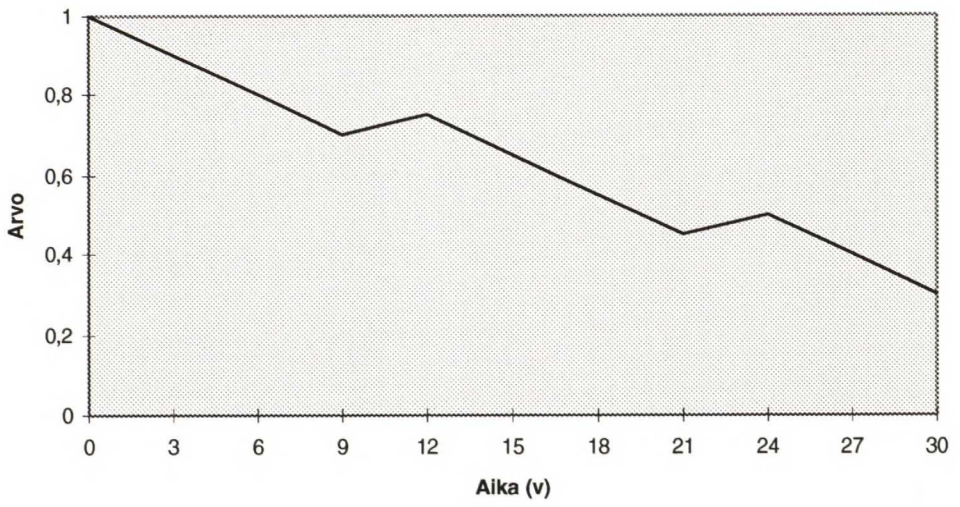
5.3.1 Mittaushuolto

Asettaa uuden arvon jollekin mitattavalle suureelle. Tuloksen vanhetessa virhearviota kasvatetaan. Mikäli tulos on vanhempi kuin kaksi kertaa annettu mittausväli, ei tulosta käytetä lainkaan laitteen kunnon arvioinnissa.

Mittaushuoltojen yhteydessä tehdään usein samalla pieniä korjauksia. Nämä pikkukorjaukset eivät kuitenkaan vaikuta merkittävästi kuntoon joten ne voidaan jättää järjestelmässä huomiotta.

5.3.2 Määräaikaishuolto

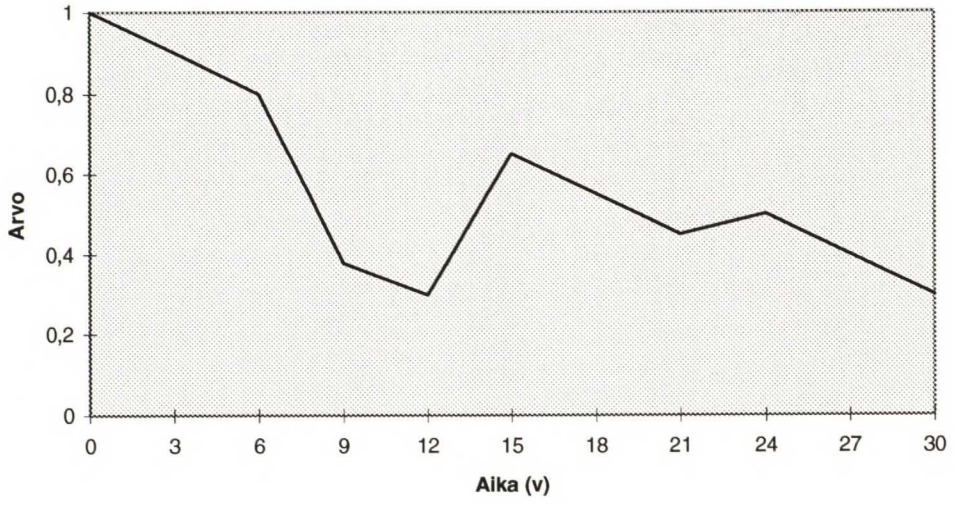
Määräaikaishuollot parantavat laitteen termistä kulumista jonkin kiinteän arvon verran, joka voi nostaa laitteen kunnon korkeintaan uuden veroiseksi eli 1:een. Mikäli laitteen alikomponenteille ei tehdä vastaavaa huoltoa, niiden huonommat kuntoarviot heikentävät päälaitteen arviota todellisuutta vastaavaksi.



Kuva 5: Esim.:Lineaarinen kuluminen ja määräaikaishuoltojen vaikutus

5.3.3 Korjaus

Korjaustoimenpide parantaa termistä kulumista jonkun lajikohtaisen vakiomäärän verran. Koska laajoja huoltoja ei tehdä, parannus ei voi nostaa kulumisarvoa senhetkistä käyräarvoa arvoa paremmaksi (mikäli se ei ole alentunut, korjaus ei vaikuta lainkaan).

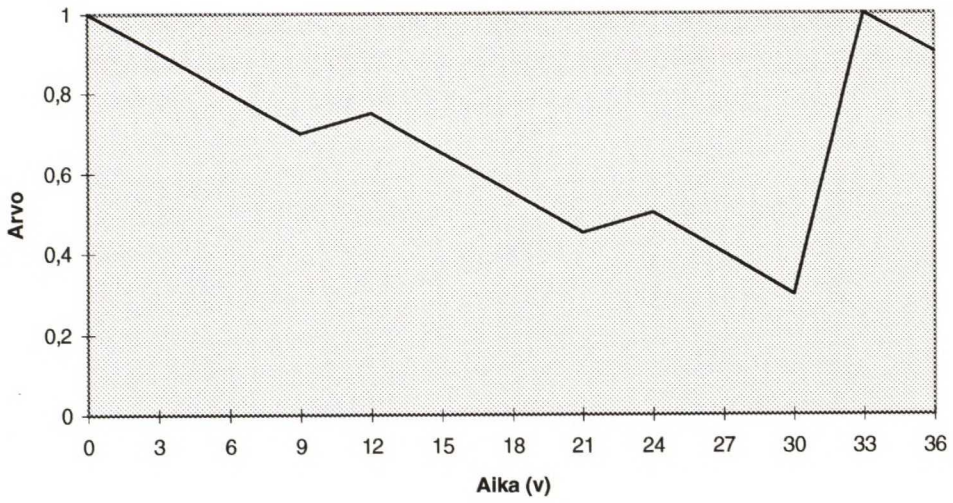


Kuva 6: Esim. lineaarinen kuluminen, korjaus- ja määräaikaishuoltojen vaikutus.

5.3.4 Vaihto

Vaihto nollaa termisen kulumisen, palauttaen sen täyteen 1:een. Kaikille laitetyypille määritellyille mittauksille annetaan paras mahdollinen arvo ja vanhat mittaustulokset pyyhitään pois.

Mikäli laitteen kaikkia alikomponentteja ei vaihdeta, niiden huonommat kuntoarviot heikentävät päälaitteen arviota. Käyttäjän itse syöttämillä mittaustuloksilla voidaan kuvata muita mahdollisesti huonommaksi jääviä komponentteja, esim. käytettyjen laitteiden vaikutusta.



Kuva 7: Esim. lineaarinen kuluminen, määräaikaishuoltojen vaikutus ja vaihto eliniän loputtua 30 vuoden jälkeen.

5.4 Kuormituksen vaikutus

Tiettyyn rajaan K_{nk}(normaalikuormitus) asti kuormitus ei vaikuta käytännössä lainkaan muuntajien kulumiseen. Siitä aina K_{mk}(maksimikuormitus) tasolle asti muuntajan kuluminen on jonkin verran nopeampaa. Maksimikuormakin voi ylittyä tilapäisesti esim. muualla verkossa tapahtuneiden häiriöiden vuoksi. Se kuluttaa laitteita kuitenkin erittäin nopeasti. Kuormitus otetaan laskennallisesti huomioon ainoastaan vanhenemisessa. Ylikuormituksen vaikutus näkyy myös mittaustuloksissa.

Kuormituksen vaikutusta arvioidaan kahdella tasolla: keskimääräisenä pitkäaikaisena (esim. 2 viikon jaksoissa) kuormituksena ja läpilyönteinä jotka kytketään irti lähes välittömästi. Pitkäaikainen kuormitus on mielenkiintoinen pääasiassa muuntajille, lyhytaikaiset ylilyönnit koskevat erityisesti eristimiä ja katkaisijoita mutta myös muuntajia.

5.4.1 Keskimääräinen kuormitus

Keskimääräinen kuormitus lasketaan laitteen läpi kulkevana tehona. Normaali- ja maksimikuormitus määritellään verkon mitoitusvaiheessa. Nämä tiedot löytyvät muuntajien ominaisuustiedoista.[1]

Jos keskimääräinen kuormitus on korkeampi kuin normaalikuormitus se vaikuttaa kulumista kasvattavasti ao. kaavalla lasketun prosenttiluvun verran.

$K_l = 2 * (\text{kuorma} - K_{nk}) / (K_{mk} - K_{nk}) * 100\%$

Esim. jos nykyinen kuntotaso on 0,7 ja uusi taso on 0,5 ja ym. kerroin% on 15%, putoaa kuntotaso 0,23 eli uusi taso onkin 0,47.

Esim. vuosina 6-12 30% ylikuormitus

5.4.2 Ylikuormitus

Hetkelliset ylikuormitukset laskevat muuntajan kuntotasoa hyvin nopeasti. Niitä syntyy erityisesti verkon vikaantuessa ja joskus myös kytkentöjen yhteydessä. Vakavammat ylikuormitustilanteet on dokumentoitava samalla kun häiriötilanteetkin. [1]

Hetkelliset ylikuormitustilanteet huonontavat suoraan muuntajan kulumisarviota. Tämä on laitetyyppikohtaisesti määriteltävä kiinteä määrä. Esimerkiksi ilmakatkaisijat ovat ylilyönnin jälkeen yleensä melko hyvässä kunnossa, sen sijaan öljykatkaisijan öljyn ominaisuudet muuttuvat eikä se enää ole kovin luotettava. Niille voidaan siis antaa esim. arvot 0,05 ja 0,25.

5.5 Historia

Muuntajille tehdään huoltoja ja tarkastuksia säännöllisen aikataulun mukaan. Siitä huolimatta niihin tulee joskus yllättäviä vikoja. Tarkastuksien yhteydessä voidaan myös havaita tarvetta ylimääräisille huolloille.

Näiden historiatietojen perusteella voidaan joskus havaita selkeitä käyttäytymistrendejä laitejoukoille. Laitejoukkojen rajausperusteina voi olla mm.

- Laitetyyppi
- Sijaintiympäristö
- Mittaukset
- Kunnossapitotyyppi

Lisäksi voidaan verrata tyypin sisällä yksittäisiä muuntajia. Muuntajissa voi olla myös yksilöllisiä eroja; jos jonkun laitteen historiassa on paljon vikaantumisia ja ylimääräisiä huoltoja, sen vaihtaminen on järkevä vaihtoehto vaikka sillä olisi laskennallista elinikää paljonkin jäljellä.

Lisäksi lähtötietojen perusteella voidaan tehdä päätelmiä eri muuntajien huoltokustannuksista, samoilla kriteereillä kuin yllä. Joidenkin muuntajien huoltoon ja ylläpitoon kuluvat kustannukset saattavat olla suuremmat kuin muilla vastaavilla, johtuen esimerkiksi hankalammista kulkuyhteyksistä. Syitä ei yleensä voida automaattisesti päätellä. Havaitseminen ja ryhmittely on tietokoneella helpompaa kuin ihmisille, syiden arvaaminen taas onnistuu ihmiseltä helpommin. Esimerkkinä voi olla hallintoalueiden väliset kohteet tai tiedostamaton priorisointi joillekin kohteille.

5.6 Mittaukset

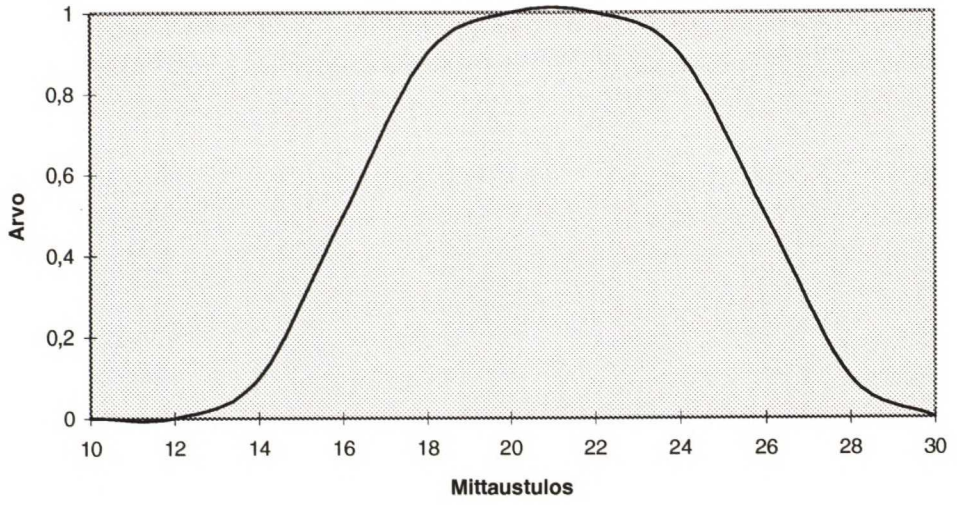
Mittaustulokset saadaan joko lukuarvona tai luokituksena (esim. Hyvä, Kohtalainen, Huono). Mittaustulosten yhteisvaikutuksesta saadaan arvio laitteen kunnosta sekä arvio tuloksen luotettavuudesta. Mittaustulokset ovat absoluuttisia arvoja jotka on suhteutettava vertailukelpoisiksi arvoiksi. Luokituksille on määriteltävä vastaava lukuarvo [0,1]. Lukuarvoisille mittauksille määritellään käyrä jonka mukaan arvo luokitellaan [0,1] asteikolle. Käyrillä on neljä perusmuotoa: arvoväli, alaraja, yläraja ja pistemäinen arvo.

Mittaustuloksilla on arvoa ainoastaan mikäli ne ovat tuoreita. Mikäli kyseessä on määräaikaismittaus, annetaan tulokselle virhearvio sen mukaan kuinka suuri osa mittausten välisestä ajasta on kulunut. Mikäli mittauksella ei ole määräaikaväliä, käytetään aikavälinä yhtä vuotta. Virhearviot vaikuttavat lopputuloksen luotettavuuteen.

Mittaustulosten virheitä ei säännöissä oteta huomioon. Mikäli systemaattisia virheitä ei tehdä, virheitä tulee yhtä paljon kumpaankin suuntaan. Sumean logiikan laskentasäännöillä tällaiset poikkeamat eivät kertaudu lopputuloksessa niin että ne johtaisivat olennaisesti oikeasta poikkeavaan loppupäätelmään. Mikäli mittaustuloksissa on systemaattista virhettä kumpaan tahansa suuntaan, ei minkäänlainen järjestelmä voi tuottaa oikeita tuloksia.

5.6.1 Arvoväli

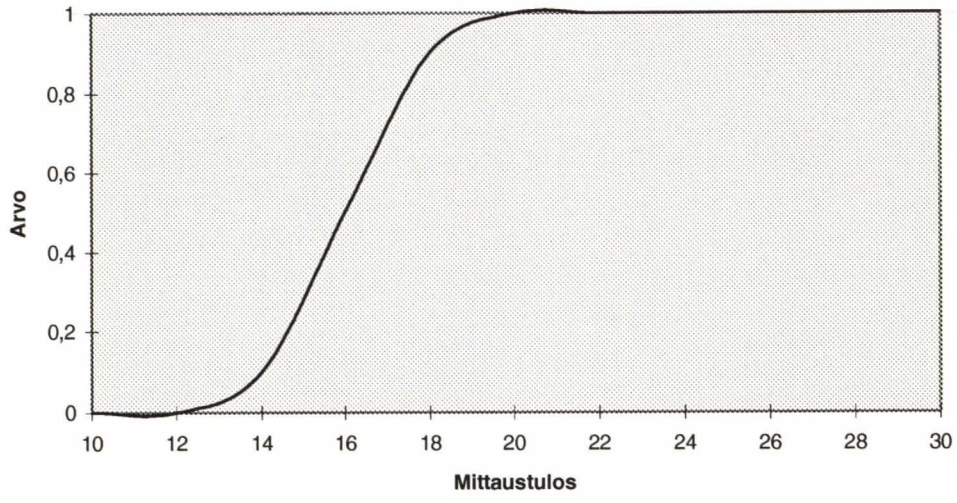
Mittaustulokselle on annettu ala- ja yläraja, joiden välissä mittaustuloksen tulisi pysyä. Keskellä mittaustuloksen arvoväliä arvo on 1, reunoilla hieman vähemmän, reunojen ulkopuolella laskee nopeasti kohti 0:aa. [6]



Kuva 8: Mittaustuloksen arvot arvovälillä 18-24

5.6.2 Alaraja

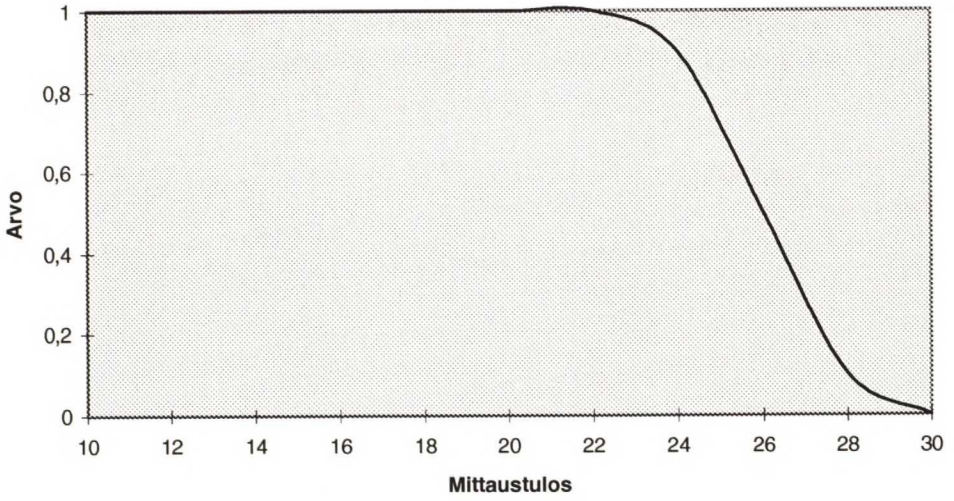
Mittaustulokselle on annettu alaraja, jota alemmas mittaustulos ei saisi laskea. Mittaustuloksen laskiessa alarajan alle arvo tippuu nopeasti kohti 0:aa. [6]



Kuva 9: Mittaustuloksen arvot, alarajana 20

5.6.3 Yläraja

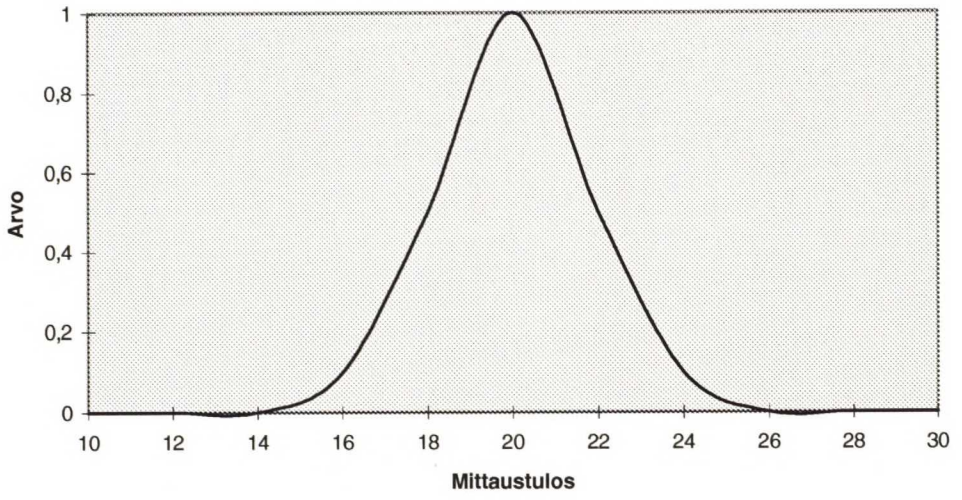
Mittaustulokselle on määritelty yläraja jonka yli se ei saisi nousta. Mittaustuloksen noustessa ylärajan yli arvo tippuu nopeasti kohti 0:aa.[6]



Kuva 10: Mittaustuloksen arvot, ylärajana 22

5.6.4 Pistemäinen arvo

Mittaustulokselle on annettu pistemäinen tavoitearvo, jonka läheisyydessä tuloksen tulisi pysyä. Tarkalleen annetussa arvossa pistemäinen mittaustulos saa arvon 1, läheisyydessä tippuu nopeasti kohti 0:aa.6]



Kuva 11: Mittaustuloksen arvot, pistemäisenä arvona 20

Käyrän muoto voi vaihdella tiettyjen perustyyppien kesken: esim. kolmiokäyrä tai normaalijakauma. Käyrä voi olla myös täysin vapaamuotoinen koska se on määritelty sumeana joukkona, esim. jompaankumpaan reunaan painottunut.

5.7 Yhteisvaikutukset

Kullakin muuntajatyypillä on sen omat mitattavat ominaisuudet jotka saavat arvoja [0,1], joko raja-arvojen tai luokitusten perusteella. Lisäksi monet kohteet koostuvat osalaitteista joiden kunnon yhteisvaikutus vaikuttaa ratkaisevasti koko laitteen kuntoon. Kaikista näistä osatekijöistä saadaan muuntajan lopullinen kuntoarvio jota käytetään toimintaehdotusten perustana.

Säännöt yhteisvaikutukselle: R3 tuottaa totuusarvon [0,1] lauseelle 'laite on hyväkuntoinen'.

R1: O_1 and O_2 and ... and O_N
R2: M_1 and M_2 and ... and M_N
R3: R_1 and R_2

O_i = osalaite numero i:n saama kunnossapitoarvo välillä [0,1]
M_i = mittaustulos numero i:stä saatu sumea arvo välillä [0,1]

Molemmat lasketaan loppuun ennen yhdistämissäännön soveltamista. Samoin R1 ja R2 lasketaan ennen R3:n käyttöä.

Sääntö on purettu kolmeen osaan vaikka siitä olisi voitu tehdä myös yksi ainoa sääntö. Sumeat and-operaatiot ovat erillään koska voidaan haluta käyttää eri and-operaatiota osalaitteiden ja mittaustuloksien yhdistämisessä. Nollatasot voivat vaikuttaa sääntöjen välissä eikä pelkästään lähtötietoihin ja lopputulokseen. Lisäksi ratkaisu edistää modulaarisuutta toteutuksessa.

Allaolevat säännöt puolestaan ovat jossain määrin enemmän sarjallisia koska esimerkiksi keskimääräisten kuntotietojen laskemiseksi on oltava tiedossa kaikkien muiden kohteiden kunto. Vaikka säännöt on pilkottu pienehköihin osiin, niistä voidaan yhdistellä joukko kokonaissääntöjä lopullisessa toteutuksessa käsittelyn helpottamiseksi; säännöt voivat laueta myös samanaikaisesti kuten yleensä sumean logiikan sovelluksissa.

Säännöissä on alla käytetty myös normaaleita aritmeettisia operaatioita joiden tulos on välillä [0,1]. Tämä on eräs tapa sumeuttaa tarkka lähtötieto. Vaihtoehtona on määritellä aina luokat joiden perusteella saadaan kyseiseen luokkaan ryhmiteltyjen tietojen vastaava sumea totuusarvo. Alla aritmeettisia operaatioita ovat ne joiden kaikki argumentit ovat kiinteitä (K_x).

5.8 Säännöt

Muuntajan määräaikaishuollon tarpeellisuus
R4: $\min((t / t_i * (1+K_l)), 1)$

Aika edellisestä määräaikaishuollosta	t
Määräaikaishuolto aikaväli	t_i
Ylikuormituskerroin	K_l

Muuntajan huollon kannattavuus taloudellisesti

R5: not min(C_r / C_c ,1) or S_p

Huollon hinta	C_r
Uuden vastaavan laitteen hinta	C_c
Laitteen (asiakkaan) prioriteetti	S_p

Muuntajatyypin huollon kannattavuus
R6: min(N_f / N_{maxf},1) and not (S_d)

vikojen lukumäärä	N_f
Suurin vikojen lukumäärä vastaavilla laitetyypeillä.	N_{maxf}
Poistettava tyyppi	S_d

Muuntajan huollon kannattavuus teknisesti
R7: not min(K_a / t_{max} ,1) and R6

Ikä	K_a
Elinikä	t_{max}

Muuntajan huollon tarpeellisuus, yhdistelmäsääntö

R8: not R3 and R4 and R5 and R7

Muuntajan vaihdon suositeltavuus, yhdistelmäsääntö

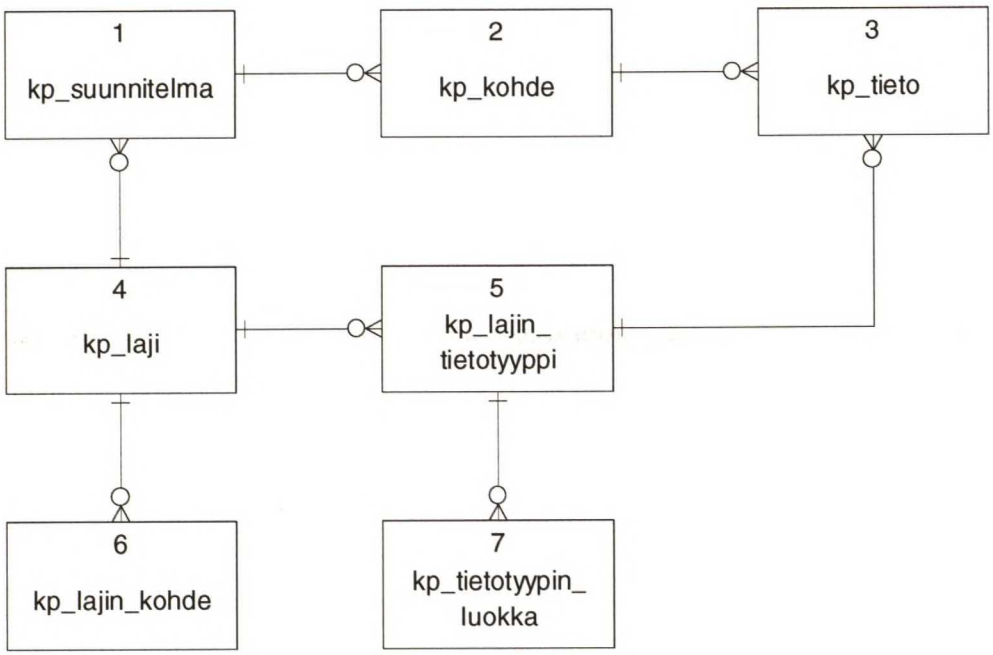
R9: not R3 and not R5 and R7

6 Toteutus

6.1 Sovelluksen tietokanta

Koko verkonhallintajärjestelmän tietokanta on laaja ja monimutkainen, eikä sen rakenneta esitetä tässä työssä. Kunnossapitosovelluksen osalta tietokannan rakenteesta kuvataan tarpeelliset osat.

Tietokannasta on olemassa muistissa tietokantaa vastaava relaatiomalli josta voidaan tehdä hakuja samankaltaisesti kuin normaalista tietokannasta sekä lisäksi geometrisia hakuja. Muistissa olevaan kantaan ladataan tiedot siitä osasta verkkoa jota käyttäjä haluaa käsitellä. Rajaus tehdään tyypillisesti geometrisella jaolla, esim. karttalehtien perusteella, mutta myös erilaisten loogisten kokonaisuuksien lataaminen on mahdollista.



Kunnossapitojärjestelmän tietokannan ER-kaavio

Osalle tauluista on erilliset historiataulut joihin talletetaan edelliset arvot uusia lisättäessä. Niistä poistetaan vanhentuneet tiedot säännöllisin väliajoin.

Seuraavassa on hahmoteltu tarpeellisia ominaisuustietoja käsiteltävien taulujen osalta. Taulujen rakenteesta on jätetty pois kaikki tehokkuuteen tai käyttömukavuuteen liittyvät osat. Rakennetta on yksinkertaistettu myös jättämällä joitakin 3NF:ään liittyviä tauluja kuvaamatta.

kp_suunnitelma
- laji_id
- status
- määräaika

kp_kohde
- suunnitelma_id
- laite_id
- laitetyyppi_id
- prioriteetti

kp_laji
- aikaväli (määräaikaistoimenpiteille)

kp_lajin_kohde
- laji_id
- laitetyyppi_id

kp_lajin_tietotyyppi
- laji_id
- luokka (arvo/luokitus/molemmat)
- edellinen (talletetaanko historiatiedot)
- pakollinen

kp_tietotyypin_luokka
- tietotyyppi_id
- id (tulee kp_tieto:n luokka-kentän arvoksi)
- selitys

kp_tieto
- kohde_id
- tietotyyppi_id
- luokka
- arvo
- kiireellisyys
- Päivämäärä

6.2 Sumean järjestelmän toteutus

Koska koko sovellusalueen tietämys ja käytännöt vaihtelevat eri organisaatioissa, on varsinaisten sääntöjen lopullinen määrittely jätettävä sähkönjakeluyhtiöille itselleen. Paras paikka tallettaa säännöt on sama kuin missä myös niiden tarvitsema data on eli tietokanta. Lajikohtaiset säännöt koostuvat ennenkaikkea mittaustulosten arvioinnista, mutta yo. säännöt kirjataan tietokantaan samoin periaattein jotta niitäkin voidaan kokemustiedon karttuessa säätää lähemmäs optimaalista. Näin niihin on myös mahdollista lisätä uusia piirteitä ilman että koodia tarvitsee muuttaa. Jotkut sumeat parametrit tarvitsevat kuitenkin ohjelmallista tukea: esim. erilaiset summa- ja keskiarvolausekkeet.

6.2.1 Sääntö

Kokonainen sääntö muodostuu yhdestä pääsäännöstä ja mahdollisesti joukosta alasääntöjä joilla voi vuorostaan olla alasääntöjä. Säännöt muodostavat puuhierarkian jota sovelletaan järjestyksessä alhaalta ylöspäin, laskien ensin tarvittavat välitulokset. Mikäli sääntö koostuu ainoastaan yhdestä osasta, juurisäännölle pätee samat periaatteet kuin normaalille lehtisäännölle. Laskenta päättyy käsiteltävän haaran osalta kun alisääntöjä ei enää löydy, vaan säännöllä on ainoastaan parametreja.

Operaatiokenttä määrää miten alisääntöjen ja parametrien tulokset yhdistetään tämän säännön tulokseksi. Se saa yleensä arvoikseen joko and tai or; mutta alimman tason säännöillä ei ole annettu operaatiota. And- ja or -operaatioita on sumeille totuusarvoille määritelty lukuisia erilaisia; operaatio-kenttä määrää mitä nimenomaista versiota käytetään tässä säännössä.

Jokaiselle säännölle tarkastetaan sen nollataso ennen tuloksen käyttöä; mikäli tulos alittaa nollatason, se tulkitaan puhtaaksi nollaksi. Mikäli käyttävä sääntö vaatii tuloksilta nollasta poikkeavia arvoja, myös koko säännön tulokseksi tulee nolla. Nollatasoksi voidaan määritellä myös 0 tai 1 jolloin tulos joko on aina merkittävä tai käytetään normaalia Boolen logiikkaa.

Sääntöä laskettaessa otetaan huomioon alasäännöillä ja parametreilla olevat painoarvot. Painoarvot ovat lukuja 0:sta 100:aan, oletuksena 100 eli täysi painoarvo. Painoarvot vaikuttavat ainoastaan lopputuloksen luotettavuuden arviointiin, totuusarvoja ei voi useimmilla and ja or -operaatioilla skaalata mitenkään.

6.2.2 Suodattimet

Suodattimet ovat sääntökohtaisia. Suodattimet määritellään tarvittaessa kaikille säännöille. Myös alisääntöihin voidaan soveltaa suodattimia, muuttaen välitulosta ennen sen käyttöä korkeamman tason säännöissä. Suodatin-taulu sisältää pelkästään viittauksen suodatintyypin ja säännön välillä, varsinaiset suodattimen tiedot ovat suodatintyyppi-taulussa.

Not-operaatio on määritelty suodattimena, ei säännön ominaisuutena koska sille ei tarvita kuin yksi argumentti. Sen vaikutus annettuun sumeaan joukkoon on samankaltainen kuin tavallisilla suodattimilla. Eri not-tyypit voidaan joutua määrittelemään jossain määrin erikoistapauksena, koska toisista suodattimista poiketen se kääntää tulosta, ei pelkästään vahvista tai heikennä sitä.

Suodattimia sovelletaan tulokseen annetussa järjestyksessä. Järjestyksellä on väliä erityisesti not-operaation kanssa mutta myös muilla suodattimilla lopputulos vaihtelee keskinäisestä järjestyksestä riippuen.

Suodattimien käyrät (paitsi not) lasketaan jollakin exponentiaalisella kaavalla jonka parametrointi vaikuttaa voimakkaasti. Aluksi käytetään kaavaa $\mu A[x]^{(100/(100\text{-vaikutus}))}$. Tarkka kaava löytyy kokeilemalla toteutuksen yhteydessä, luultavasti tekijä (100/(100-vaikutus)) korotetaan johonkin potenssiin vaikutuksen korostamiseksi. Muunlaisia suodattimia voidaan määritellä tarpeen mukaan.

Vaikutus on suodatintyyppikohtainen arvo joka ratkaisee suodattimen vaikutuksen voimakkuuden. Se annetaan prosenttilukuna. Vaikutusarvo on skaalattu s.e. 0 merkitsee 'ei vaikutusta, 100 normaalia Boolen logiikkaarvoa ja etumerkki säätää loiveneeko vai jyrkkeneekö arvojen käyrä (-100 ei vaikuta kuin neliöjuuren verran, mikä on luultavasti liian rajoittunut). Seuraavassa on lueteltu joukko määriteltäviä suodattimia vaikutusarvoineen. Vaikutusten kuvaus on jätetty pois, olettaen että termit ovat jossain määrin itseään selittäviä. Suodattimilla ei ole arvoihin liittyviä vaikutuksia vaikka niiden nimet niin antavat olettaakin; about, above, below, yms. suodattimilla on lähinnä tarkoitus saada paremmin kuvaava vaikutus arvovertailuissa kuin mitä yleisillä very tai slightly-suodattimilla saadaan.

- Somewhat -30%
- Slightly -10%
- Quite +20%
- Very +50%
- Extremely +80%
- About -10%
- Vicinity -20%
- Close -30%
- Precisely +50%
- Above +20%
- Below +20%
- Not

6.2.3 Parametri

Parametri on yksittäinen sääntöön vaikuttava termi. Kuhunkin sääntöön voi kuulua 0 tai useampia parametrejä. Parametrien saamat arvot yhdistetään säännön operaatiolla kuten alasääntöjenkin tulokset. Säännöllä voi olla sekä alasääntöjä että parametrejä.

Parametrin tyyppi voi olla joku seuraavista:

- Arvo
- Luokka
- Jatkuva luokka
- Lukumäärä
- Summa
- Keskiarvo
- Vertailuoperaattorit $<$, $\leq$, $>$, $\geq$, $=$ ja $\neq$

Arvo saa tulkintansa suoraan annetun taulun annetun kentän perusteella. Parametrissa määritellään sille raja-arvot tai pistemäinen arvo ja niiden perusteella parametrille lasketaan sitä vastaava sumea totuusarvo vakiolaskentasääntöjen perusteella. Mikäli arvon saamaa käyrää halutaan painottaa enemmän johonkin suuntaan, on säännölle määriteltävä suodatin joka aikaansaa halutunlaisen vaikutuksen.

Luokka saa samoin tulkintansa annetusta taulun kentästä, mutta laskennallisen arvon sijasta kyseinen kenttä on sisällöltään joku Luokka-taulussa määritellyistä Id-arvoista. Jokaista näin määriteltyä arvoa vastaa yksi kappale kiinteitä sumeita totuusarvoja.

Sellainen kentän arvo joka ei esiinny Luokka-taulun määrittelyissä tulkitaan aina 0-totuusarvoksi. Luokka-tyyppinen parametri on diskreetti vaikka se onkin sumea. Se voi saada kuitenkin saada rajattoman monta totuusarvoa.

Jatkuva luokka on erikoistapaus normaalista luokka-tyypistä. Se on määritelty jatkuvaksi, ei diskreetiksi; järjestelmä hakee automaattisesti lineaarisella regressiolla kutakin arvoa vastaavat totuusarvot jos ne eivät osu mihinkään määritellyistä luokista. Luotettavan toiminnan varmistamiseksi on totuusarvoille 0 ja 1 oltava määritellyt luokat, jotka ovat raja-arvoina luokkatyypeille. Mikäli annettu luokka-arvo on suurempi tai pienempi kuin yksikään määritellyistä luokista, käytetään äärimmäisen sen pään luokan arvoa. [3]

Lukumäärä-tyyppinen parametri hakee sellaisten kohteiden lukumäärän joiden kentän arvo on kenttäarvo-kentän määrittämä. Lukumäärästä saadaan sumea totuusarvo laskemalla samoilla periaatteilla kuin arvo-tyyppisessä parametrissa, eli saatua lukumäärää verrataan alaraja-, yläraja- ja pistearvo-kenttiin.

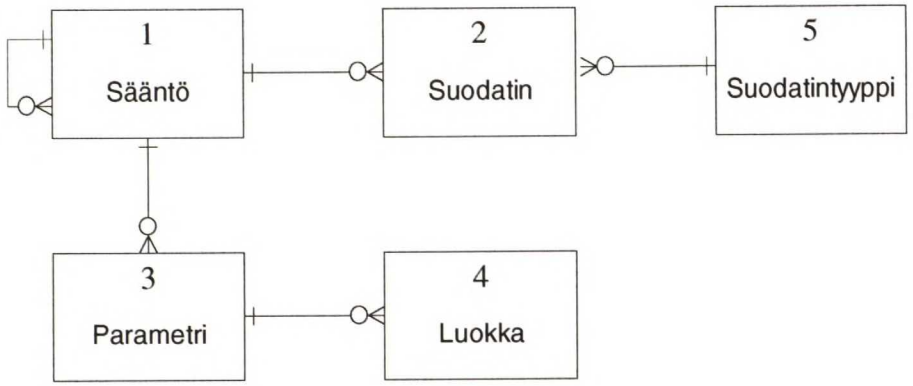
Summa- ja keskiarvoparametrit toimivat samankaltaisesti lukumäärän kanssa. Kenttäarvo-kenttää ei tarvita mihinkään; summa-parametrissa annettuja raja-arvoja verrataan suoraan annetun kentän arvojen summaan, keskiarvo-parametrissa summa jaetaan saatujen arvojen lukumäärällä ennen vertailua.

Vertailuoperaattorit on tarkoitettu parametrihierarkioiden käsittelyyn. Toisin kuin sääntöhierarkioissa, parametrien hierarkian rakenne on yksitasoinen binääripuu; kullakin parametrilla voi olla ainoastaan kaksi aliparametria. Näistä pienemmällä Id:llä olevaa verrataan annetulla operaattorilla suuremmalla Id:llä olevaan. Vertailuoperaattoreita ei tarvita normaaleille parametreille koska niillä on kiinteästi määritellyt raja-arvot joille vastaavat vertailut tehdään oletusarvoisesti. Esimerkiksi pistearvo vastaa yhtäsuuruusvertailua, sen komplementti erisuuruusvertailua, alaraja-arvo vastaa suurempi-kuin-vertailua kentän arvon ja raja-arvon välillä.

Mikäli säännössä halutaan verrata jonkun tietyn kohteen kentän arvoa esimerkiksi kentän keskiarvoon, on parametrista muodostettava hierarkinen malli jossa ylimmän parametrin tyyppi on joku vertailuoperaattoreista. Alatyypit voivat olla periaatteessa minkä tyyppisiä tahansa; on mahdollista esimerkiksi verrata kentän summaa sen keskiarvoon. Tyypillinen tapaus lienee kuitenkin kentän arvon vertaaminen keskiarvoon yms. Parametrihierarkioita ei voi tehdä kuin kaksitasoisiksi koska parametrien laskennasta tulee totuusarvoja joita ei ole mielekästä verrata vertailuoperaattoreilla. Säännön parametreista voidaan tunnistaa aliparametrit Parametri_id:n perusteella; ellei niin erityisesti määritellä, aliparametrien oma totuusarvo ei tule mukaan säännön lopputuloslaskentaan.

6.2.4 Käsitemalli

[3] Jatkuvaa luokkaa voi hyödyntää esimerkiksi kunnossapidon ikä-käsitteen mallintamisessa. Aika on jatkuva suure mutta sen vaikutus vaihtelee; sille määritellään usein useampia arvoja mutta ensimmäinen on aina 1 ja viimeinen 0.



ER-kaavio sumean logiikan sääntöjen määrittelyyn tarvittavista rakenteista

Sääntö

- Id
- Isäsääntö
- Operaatio (and tai or[4])
- Nollataso (raja, jota pienempi tulkitaan 0:ksi)
- Kuvaus
- Painoarvo

Suodatin

- Sääntö_id
- Järjestysnumero
- Suodatintyyppi_id

Suodatintyyppi

- Id
- Kuvaus
- Tyyppi (hyvin, melko, suunnilleen, not, jne.)
- Vaikutus

Parametri

- Id
- Kuvaus
- Sääntö_id
- Parametri_id
- Tyyppi
- Painoarvo
- Taulu
- Kenttä
- Alaraja
- Ylaraja

[4] Tämä kenttä määrittelee myös käytettävän tyypin. Esimerkiksi and-operaatiossa voidaan käyttää minimiä, tuloa, keskiarvoa, Yagerin and:iä ja monia muita. Eri operaatiot vaativat kuitenkin ohjelmallisen tuen, jolleivät ne koostu jostain olemassaolevista tyypeistä jolloin ne voidaan korvata niistä muodostuvalla rakenteella.

- Pistearvo
- Kenttäarvo

Luokka
- Id
- Parametri_id
- Tunnus
- Arvo

6.2.5 Algoritmi

Sumean logiikan käsittely muodostaa alasysteemin johonkin varsinaiseen sovellukseen (esim. kunnossapito). Sen algoritmit ja käsittely on kuvattu yleisellä tasolla täällä, jotta yleistäminen muunkaltaisiin sovelluksiin onnistuisi helpommin. Kaikki tähän osasysteemiin liittyvä ohjelmakoodi on datalla ohjattavaa, varsinainen sovellusosa ei tarvitse mitään muutoksia toiminnallisuuteen. Tähän systeemiin liittyvien taulujen käsittely tapahtuu tietysti sovelluskohtaisesti. Taulujen käsittely on pääasiallinen kommunikaatiomenetelmä: muu kommunikaatio rajoittuu lähinnä laskennan käynnistämiseen ja tulosten saamiseen sääntötasolla. Alla on määritelty algoritmin rakenne jolla yleiset laskentatulokset saadaan aikaan sekä esimerkkikoodia ja -dataa jolla saadaan yhdelle kohdetyypille laskettua kuntoarvio. Esimerkki lähtötietoineen ja ajoesimerkkeineen on kokonaisuudessaan liitteessä 1. Algoritmi kokonaisuudessaan löytyy liitteestä 2.

Sääntöhierarkian läpikäynti:
1. Haetaan pääsääntö sovelluksen antamilla kriteereillä. Tämä on liittymä varsinaiseen sovellukseen päin: syötteenä saadaan tieto mitä sääntöä sovelletaan ja mille kohteelle.
2. Haetaan säännön alasäännöt rekursiivisesti kunnes saavutetaan hierarkian lehtisäännöt. Hierarkia käydään läpi syvyys-ensin-hakuna.
3. Lehtisäännöistä alkaen lasketaan säännöille yksi kerrallaan arvot pohjautuen niiden parametreihin ja mahdollisien alasääntöjen arvoihin.
4. Yhdistetään alasääntöjen arvot.

Yksittäisen säännön laskenta:
1. Haetaan säännön parametrit.
2. Kullekin parametrille tutkitaan sen tyyppi ja muunnetaan lähtötiedot sumeaksi totuusarvoksi tyyppikohtaisella toiminnallisuudella.
3. Nollatason tarkastus: Sellaiset arvot, jotka ovat alle säännössä määritellyn nollatason, asetetaan nollaksi. Mikäli säännölle ei ole asetettu omaa nollatasoa, verrataan yleiseen nollatasoon.
4. Suodattimien soveltaminen annetussa järjestyksessä.
5. Talletetaan säännön laukeamistieto eli kuinka suuren totuusarvon se lopulta saa. Tieto talletetaan ainoastaan kerran kaikkien suodattimien jälkeen.

6.2.6 Esimerkki

Sääntö

Id	1
Isäsääntö	0

Operaatio	ZADEH_AND
Nollataso	0,2
Kuvaus	R1: Kuntoarvio, parametrit
Painoarvo	1.0

Parametri

Id	2
Kuvaus	Ylimenovastus μΩ
Sääntö_id	1
Parametri_id	0
Tyyppi	VALUE
Painoarvo	1
Taulu	kp_tieto
Kenttä	arvo
Alaraja	0.0
Ylaraja	200.0
Pistearvo	0.0
Kenttäarvo	0.0

1. Haetaan pääsääntö: id 1.
2. Haetaan alasääntöjä sille: ei löydy.
3. Haetaan parametreja sille: löytyy.
4. Parametri tyyppi on VALUE.
5. Haetaan argumenteista kp_tieto-taulun rivi ja siitä kenttä arvo.
6. Verrataan sitä ylärajaan 200: arvo 179 on pienempi, joten saatu tulos on 1.0.
7. Haetaan lisää parametreja: ei löydy.
8. Yhdistetään parametrien tulokset Zadeh:in and:illä: ainoastaan 1, joten lopputulos on 1.0.
9. Yhdistetään parametrit ja alasäännöt: alasääntöjä ei ole joten tulos on suoraan parametrien tulos: 1.0.
10. Tarkastetaan nollataso: 1.0 > 0.2 joten tulos on 1.0.
11. Haetaan suodattimia: ei löydy.
12. Talletetaan laukeamistieto.
13. Palautetaan lopputulos: 1.0

6.2.7 Lopputuloksen perustelu

Säännöille talletetaan aina laskennan valmistuttua tieto siitä mikä lopputulos oli ja mikä on säännön saama painoarvo seuraavalla tasolla. Parametreille talletetaan vastaavat tiedot: niillä on määritelty oma itsenäinen painoarvonsa. Näiden välitulostietojen ja säännön rakenteen avulla on mahdollista tuottaa käyttäjälle listaus siitä mihin sääntöihin ja parametreihin saatu lopputulos pohjautuu.

Säännöille, parametreille ja suodattimille on määritelty toiminnallisten tietojen lisäksi myös kuvaus-kenttä jota voidaan käyttää juuri tämän perustelun tuottamiseksi. Jonkinlainen selkokielinen selostus asioiden merkityksestä on välttämätön. Lisäksi säännöillä ja parametreillä on painoarvo-kenttä joka antaa lisätietoa tuloksen luotettavuudesta.

Perustelulistaus lähtee päinvastaisesta järjestyksestä kuin varsinainen laskenta: ensimmäisenä näytetään ylimmän tason sääntö, sen saama arvo ja sanallinen kuvaus

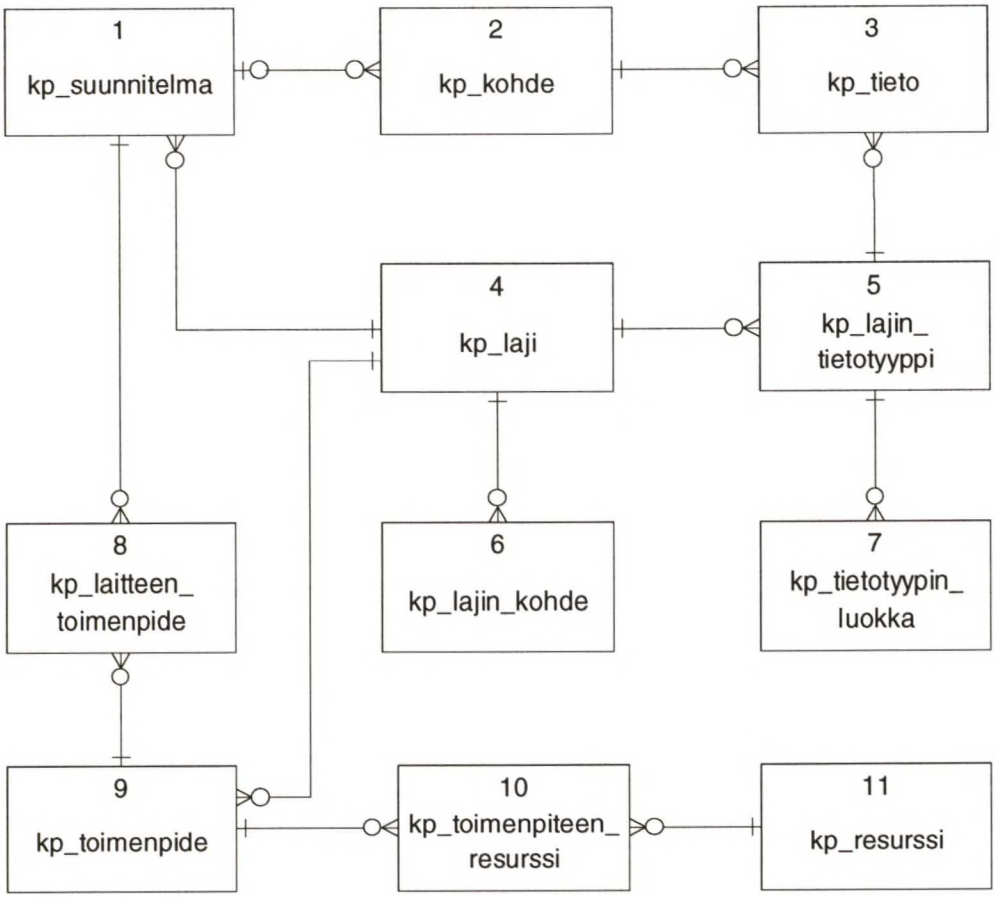
lopputuloksen luotettavuudesta pohjautuen tulokseen ja sen osien painoarvoihin. Sen jälkeen listataan vastaavat tiedot kaikista sen ensimmäisen tason komponenteista. Vasta kun kaikki ko. tason alisäännöt tuloksineen on listattu, edetään alemmalle tasolle.

Listauksen formaatti (esimerkiksi):

Kuvaus:	Onko laite hyväkuntoinen ?	
Tulos sanallisesti:	Ei	(selkeä ei, ei, ehkä, kyllä, selkeä kyllä)
Tulosarvo:	0,27	
Painoarvo:	1	(päätasolla)
Arvot:	Tulos	Arvo
Vastus	210	0,85
Toiminta-aika auki	37	0,6
Toiminta-aika kiinni	162	0,41
Alisäännöt:		
Kuvaus:	Onko kosteusvaurioita ?	
...		

6.3 Kunnossapidon toteutus

Kunnossapidon tietokantarakenne vaatii vain vähän muutoksia jotta sumea versio on toteutettavissa. Suurin osa uusista tiedoista joita säännöt vaativat talletetaan sumean järjestelmän yleisiin tauluihin. Jotkut säännöt vaativat kuitenkin tietoja joita nykyisessä tietokannassa ei ole.



Uusi kunnossapidon ER-kaavio

Järjestelmään tarvittavien uusien taulujen rakenne on kuvattu alla. Koko tietosisältöä ei ole määritelty näillekään koska käyttöliittymään, normalisointiin yms. liittyvät kentät ja taulut ovat tämän työn kannalta epärelevantteja.

kp_toimenpide
- Id
- Tyyppi
- Laji
- Tunnus
- Kestoaika
- Kustannukset

kp_laitteen_toimenpide
- Laite_id
- Laitetyyppi_id
- Suunnitelma_id
- Toimenpide_id
- Päivämäärä

kp_toimenpiteen_resurssi
- Toimenpide_id
- Resurssi_id

- Määrä

kp_resurssi
- Id
- Tyyppi[5]
- Tunnus
- Määrä

kp_laitetyyppi
- Laitetyyppi_id
- Hinta
- Poistettava

6.3.1 Tietojen talletus

Järjestelmän laitteille on talletettu niiden ominaisuustiedot. Lisäksi niille talletetaan aina kunnossapitotoimenpiteet jotka niille on tehty, mittaustulokset ja vikatiedot. Laitetyyppikohtaisesti niillä on määriteltynä muitakin tietoja kuten hinta ja kunnossapitoaikavälit.

Näistä tiedoista on aiemmin määriteltyjen tietojen perusteella mahdollista tuottaa kaikki tarvittavat tiedot mitä kunnossapidon automaattiseen suunnitteluun tarvitaan. Osa näistä tiedoista on kuitenkin sellaisia että niiden laskeminen vaatii suuria tietokantahakuja tai raskaita laskentaoperaatioita. Esimerkiksi kohteen vanhenemisesta laskettava pohjakuluminen otetaan huomioon paitsi peruskäyrä iän perusteella, myös laitteen kuormitus, olosuhteet, huollot, mittaustulokset jne. Joillakin säännöillä tarvitaan kaikkien tietyntyyppisten laitteiden keskiarvoja tai muita useisiin kohteisiin pohjautuvia tuloksia. Suuren hierarkian ylimmillä kohteilla kaikkien alakohteiden laskeminen aina kun laitteen kuntoa halutaan arvioida voi olla hyvin työlästä.

Kaikkien näiden tietojen laskeminen tarvittaessa olisi hyvin hidasta ja tehotonta, erityisesti koska tuloksia tarvitaan useimmiten suurille joukoille kohteita kerrallaan, harvemmin yksittäisille kohteille. Näinollen on perusteltua tallettaa osa tiedoista kp_kohde-tauluun turhan laskennan välttämiseksi. Tiedot voidaan tallettaa aina erityisen laskennan yhteydessä tai voidaan tehdä säännöllisesti ajettava eräajo-ohjelma joka laskee tarvittavat tiedot silloin kun muu käyttö on vähäistä. Kunnossapitotiedot muuttuvat yleensä melko hitaasti ja silloin kun ne muuttuvat nopeasti, on yleensä kyse vikatilanteesta tai muusta poikkeuskäsittelyä vaativasta tilanteesta. Niinpä tällainen redundantin tiedon talletus ei aiheuta merkittäviä eheysongelmia.

6.3.2 Kunnossapitoarvioiden generointi

Työlista liittyy aina kunnossapitosuunnitelmaan. Kunnossapitosuunnitelma voi sisältää toimenpiteitä rajattoman monelle kohteelle, jotka voivat olla rajattoman montaa eri tyyppiä. Käytännössä kuitenkin rajoitutaan korkeintaan muutamalle eri tyyppiselle

[5] Tyypillisiä resursseja ovat mm. työvoima, maansiirtokoneet, kuljetusvälineet, työkalut.

kohteelle tehtäviin huoltoihin. Mikäli kohdetyyppejä on enemmän, ne yleensä liittyvät johonkin korkeamman tason kohteeseen jolle huolto voidaan määritellä. Kunnossapitosuunnitelmat millään aikavälillä eivät saa mennä päällekkäin.

Työlistan generointi aloitetaan hakemalla kp_lajin mukaiset laitetyypit joille kunnossapitosuunnitelma tehdään. Laitetyyppien perusteella haetaan kaikki kyseiseen laitetyyppiin liittyvät kohteet tietokannasta, rajaten niitä annetuilla ehdoilla. Käyttäjän antamat lisäehdot voivat olla tyyppiä

- Edellisesti huollosta kulunut aika
- Vikatyyppi
- Laitteen ominaisuustieto/tiedot[6]
- Alue

Löydetyille hakuehdot täyttäville kohtelle lisätään kp_kohde-rivi mikäli niillä ei sellaista jo ole. Rivi on suunnitelmakohtainen ja kullakin laitteella voi olla vain yksi kp_kohde per suunnitelma. Kohdejoukkoa voi kuitenkin halutessa täydentää, myös sen jälkeen kun suunnitelman toteutus on jo alkanut.

Seuraavaksi lasketaan kuntoarvio kaikille niille kp_kohde-riveille joille sitä ei ole aiemmin laskettu. Laskenta aloitetaan vakioikääntymisellä, koska se tulee laskettavaksi joka tapauksessa, vaikka kohteelle ei mitään muuta tulisikaan. Vakioikääntyminen on luultavasti myös suurin suorituskykyyn vaikuttava tekijä koska siinä tehdään runsaasti tietokantahakuja ja se tehdään jokaiselle kohteelle, mikä voi hierarkisissa kohteissa merkitä suurtakin määrää käsiteltäviä kohteita. Algoritmi kaivannee siis käytännön toteutuksessa voimakasta optimointia.

1. Haetaan kohteen ikä ja elinikä sen ominaisuustiedoista
2. Haetaan laitetyypin ikääntymiskäyrä parametri-taulusta
3. Etsitään ikääntymiskäyrästä ne arvot jotka lähimmin ympäröivät kohteen ikää.
4. Lasketaan lineaarisesti niiden välistä tarkemmin ikää vastaava kulumisarvo
5. Haetaan kp_laitteen_toimenpide-taulusta kohteelle mahdollisesti aiemmin tehdyt huoltotoimenpiteet
6. Lasketaan huoltotoimenpiteiden vaikutus kohteen kuntoarvioon
7. Haetaan kp_tieto-taulusta kohteella mahdollisesti olleet ylikuormitusjaksot
8. Lasketaan ylikuormitusjaksojen vaikutus. Periaatteessa on mahdollista että ylikuormitus vie kohteen kuntoarvion negatiiviseksi. Tämä kuitenkin poistetaan ennen seuraavia laskentavaiheita asettamalla arvo 0:aan.

Seuraavaksi lasketaan laitteiden alikohteiden kunnossapitoarviot. Laskennassa ei vaadita niillä olevan muita laskettavia kunnossapitotietoja kuin ajallinen kuluminen. Muutkin tiedot lasketaan kuitenkin tarvittaessa rekursiivisesti aivan samoilla periaatteilla kuin päätason kohde, mukaanlukien hierarkiassa vielä alempana olevien kohteiden laskenta.

---

[6] Laitteen ominaisuustietojen perusteella tehtävä yleinen rajaus on tehtävissä siten että käyttäjä tarvitsee vain valita listalta mitä kenttiä hän haluaa rajata ja millä arvoilla. Hierarkisten kyselyiden tekeminen vaatisi jo lähes sql:n veroisen kyselykielen kehittämistä mikä ei kuulu tähän työhön.

1. Haetaan kohteen kaikki alakohteet. Hierarkia on mallitettu kohteen ominaisuustiedoissa s.e. alakohde viittaa pääkohteeseen id:llä.
2. Lasketaan kaikkien kohteiden kunnossapitotiedot rekursiivisesti samoilla periaatteilla kuin päätason kohde.
3. Yhdistetään alikohteiden tulokset

Kunnossapitolajille voi olla määritelty lukuisia mittaustyyppejä. Seuraavaksi käydään läpi kaikki parametri-taulussa määritellyt mittaukset ja kohteella mahdollisesti olevat niitä vastaavat mittaustulokset kp_tieto-taulussa. Jokaiselle mittaukselle tarkastetaan ennen käyttöä ettei se ole vanhentunut. Jokaisesta kohteen mittaustyypistä otetaan ainoastaan tuorein versio, mikäli mittaustuloksia on useita.

1. Haetaan kunnossapitolajin mukaiset mittaukset
2. Haetaan mittauksia vastaavat tulokset kohdekohtaisesti
3. Valitaan kustakin mittaustyypistä viimeisin mittaustulos
4. Tarkastetaan ettei mittaustulos ole vanhentunut
5. sovelletaan parametri-taulussa määriteltyjä raja-arvoja tai parametrin luokkia tulokseen ja saadaan sille sumea arvo.
6. Yhdistetään mittaustulokset
7. Yhdistetään mittaustulokset ja alikohteet

Hierarkioiden läpikäyminen vaatii runsaasti tietokantahakuja. Järjestelmässä käydään läpi kolmenlaisia hierarkioita: sääntöjen, parametrien ja käsiteltävien kohteiden hierarkioita. Varsinkin sääntöjen osalta on syytä ladata ne heti aluksi keskusmuistiin, jotta niiden rekursiiviseen läpikäymiseen kuluu mahdollisimman vähän aikaa. Käsiteltävien kohteiden lukumäärä ei aina anna mahdollisuutta niiden muistiin lataamiseen. Esimerkiksi jos kohteen hierarkia on 4-tasoinen ja alikohteita on keskimäärin 2 kappaletta, tarvitaan kunkin käsiteltävän kohteen laskemiseksi tällöin keskimäärin 1+2+4+8=15 tietokantahakua ja säännön suoritusta. Indeksoinneista ja muista tietokannan optimoinneista huolimatta suorituskyky on selkeästi suurilla kohdejoukoilla ongelma.

Ratkaisuna tähän voisi olla kaksijakoinen toimintatapa: pienillä kohdejoukoilla ladataan ne muistiin tätä käsittelyä varten (tämä on tehtävissä tehokkaammin massakyselyinä kuin kohdekohtaisina hakuina) ja suurilla kohdejoukoilla hoidetaan laskenta eräajona. Jos em. esimerkkiä jatkaen laskettaisiin 10000 kohteelle kunnossapitoarvo, keskimäärin 0,1s/haku, kokonaisajaksi tulee 15000 sekuntia eli hieman yli 4 tuntia.

7 Paikkatiedonhallinta ja sumea logiikka

Kunnossapidon suunnittelussa otetaan usein huomioon myös sijaintipohjaisia riippuvuuksia. Verkonhallintaohjelmistoissa sijaintitiedolla on erittäin keskeinen rooli. Sumealla logiikalla on periaatteessa mahdollista kuvata sellaisia paikkatiedonhallinnan riippuvuuksia joiden kuvaaminen tavanomaisella logiikalla on vaikeaa. Tällaisia ovat esim. useasta tekijästä koostuvat riippuvuudet.

Sumeaa logiikkaa ei ole kirjallisuudessa sovellettu paikkatiedonhallintaan lainkaan vaikka sumeat käsitteet ovat paikkatiedossa hyvin yleisiä. Esimerkiksi käsitteet lähellä, vieressä, takana ovat luonteeltaan epämääräisiä eikä niille ole mahdollista vetää mitään tarkkoja,

yleispäteviä raja-arvoja. Ne ovat kuitenkin intuitiivisesti selkeitä, joten niille pitäisi voida olla määritelmä. Sumeaa logiikkaa ei ole laajemmassa mittakaavassa käytössä myöskään kaupallisissa GIS-sovelluksissa. Tässä työssä on kehitelty suuntaviivoja kuinka sumeaa logiikkaa voisi soveltaa yleisesti GIS-järjestelmissä ja erityisesti relaatiotietokannan päälle rakennetuissa paikkatieto- ja verkkotietojärjestelmissä. Näiden sääntöjen hyödyntäminen esimerkiksi kunnossapidossa on mahdollista sääntötasolla, mutta näiden sääntöjen rakenne eroaa jonkin verran aiemmin kuvatusta. Tämä vaikuttaa jonkin verran myös kunnossapidon sääntöjen rakenteeseen.

7.1 Alueet

Monet laskentoihin vaikuttavat tekijät kunnossapidossa (ja muillakin sovellusalueilla) ovat ilmaistu sijainnin perusteella, esimerkiksi 'ympäristö on kostea' tai 'alueella on paljon lintuja'. Tällaisista lauseista on vaikeaa ellei mahdotonta antaa mitään tarkkoja arvoja. Aina ei välttämättä ole olemassa edes suuretta jota arvioida, puhumattakaan tarkasta raja-arvosta jonka jälkeen lause on täysin totta ja jota ennen se on kokonaan epätosi.

Tällöin voidaan kuitenkin antaa em. tyyppisille lauseille totuusarvo arvovälinä [0,1], joskin tämä on usein saatavissa ainoastaan ihmisten antamana arviona. Arvio joka saadaan on aina aluekohtainen; käsiteltävä alue on jaettava halutulla tarkkuudella ja syötettävä arvot kullekin alueelle. Usein tätä ei tarvitse tehdä käsin, mikä olisi vähänkin isommalle alueelle hyvin työlästä ja kallista, vaan tiedot ovat johdettavissa muista vastaavista lähteistä. Esimerkiksi kosteus voidaan määritellä olevan sitä suurempi mitä lähempänä merta ollaan *tai* maaperän soisuus, joka on saatavissa erilaisista maastomalleista. Lintujen määrää voidaan arvioida niitä houkuttelevien kohteiden läheisyydestä, esim. kaatopaikat tai satamat.

7.2 Läheisyys

Käsite lähellä voidaan määritellä suoraan raja-arvona jota etäämmällä oleville kohteille sen totuusarvo pienenee kohti nollaa. Tämä ei kuitenkaan ota huomioon maaston muotoja, teitä yms. jotka yleensä ovat ratkaisevia reaalimaailman lähellä-käsitteessä. Yo. Luokitussysteemillä voidaan määritelllä maastolle helppokulkuisuusarvoja (esimerkiksi pelto 0,9, suo 0,5, järvi 0) ja tarkastaa välissä olevista maa-alueista helppokulkuisuuden yhteisarvo. Pidemmillä matkoilla tämä käy epäkäytännölliseksi koska reittivaihtoehtojen määrä kasvaa suureksi. Tällöin ei kuitenkaan olla edes matkan puolesta lähellä, joten kulkukelpoisuuden arviointi on tarpeetonta.

Maastossa olevia luonnollisia ja ihmisen aikaansaamia kulkureittejä (kuten metsään hakattu suurjännitejohdon kulkulinja tai taajama-alueen metsissä olevat polut) on vaikea ottaa huomioon muuten kuin erikoistapauksina. Tällaisia ei ole yleensä kattavasti dokumentoitu. Tällaiset tekijät, silloin kun ne ovat tunnettuja, voidaan ottaa huomioon alueiden kulkukelpoisuusarvossa.

Kulkukelpoisuuden arviointi liittyy maastossa kulkemiseen, siis jalan, moottorikelkalla tms. Autoliikenteessä liikkuminen on pääsääntöisesti tehtävä teitä pitkin. Tämä on normaali reitinhakuongelma tieverkoston muodostamassa suuntaamattomassa graafissa.

Teille voidaan antaa sumeita painoarvoja suhteessa niiden hyväkuntoisuuteen[7], sillä tien laatu vaikuttaa voimakkaasti kulkunopeuteen. Haun yhteydessä on annettava joku minimitaso tien laadulle, riippuen ajoneuvon laadusta.

7.3 Ryhmittelyt

Sijaintitiedon perusteella halutaan usein ryhmitellä kohteita loogisiksi kokonaisuuksiksi silloinkin kun varsinaista relaatiota ei ole olemassa. Usein normaalissa puhekielessä esiintyviä ilmaisuja ovat esimerkiksi 'keskustassa' tai 'pääkaupunkiseudulla' joille ei ole määriteltävissä mitään aivan tarkkaa rajaa jonka jälkeen ilmaisu ei enää päde. Kaikille tällaisille ryhmittelyille ei ole löydettävissä mitään ohjelmallisesti havaittavaa yhdistävää tekijää, mutta joillekin se on mahdollista. Esimerkiksi voidaan saada selville kohteet jotka ovat 'Päijänteen rannalla' tai 'Turun keskustassa' mutta ym. 'pääkaupunkiseutu' ei ole mahdollinen ilman erityismäärittelyitä koska alueella on niin monenlaista maastoa (ja koska se tarkoittaa aivan eri asiaa helsinkiläiselle kuin keravalaiselle). Tarkoilla aluerajamäärittelyillä on tietysti mahdollista määritellä alue kuin alue.

Suoran sijannin ohella voidaan kohteita ryhmitellä maaston ominaisuuksien perusteella, jonkin tietyntyyppisen kohteen läheisyydellä ja suhteessa toisiinsa. Tällöin yleensä rajataan joku tietty alue jota tarkastellaan tai lähdetään liikkeelle jostain tietystä tarkasteltavasta kohteesta. Esimerkiksi voidaan rajata paperitehtaiden läheisyydessä olevia kohteita jolloin lähdetään liikkeelle rajatulla alueella olevista paperitehtaista, tai sähkölinjalla olevat kohteet jolloin lähdetään liikkeelle solmupisteestä josta linja alkaa, kulkien aina seuraavaan pylvääseen.

7.4 Säännöt

7.4.1 Paikkatietosäännöt, maastomallisäännöt

Kahden kohteen lähekkäisyys maastossa

R18: $MAX_{path}(\text{not } \min(\Sigma(S_t),1))$
Suurin mahdollisten polkujen[8] saamista arvoista MAX_{path}
Maastoruudun hankalakulkuisuus S_t

Kahden kohteen lähekkäisyys teitä pitkin

R19: $\text{not}(MIN_{path}(\min(\Sigma(K_l*K_r),1)))$
Pienin mahdollisten polkujen saamista arvoista MIN_{path}
Tieosuuden pituus / km K_l
Tieosuuden hankalakulkuisuus[9] S_r

---

[7] Teille määritellään itse asiassa ainoastaan tieluokka, josta voidaan johtaa kaikki muut tiedot. Tiestön suuren määrän vuoksi muu ei ole mahdollista, eikä muunlaista dataa ole saatavissa valmiina.

[8] Sellaiset polut joiden ruutujen hankalakulkuisuuksien summa jää alle yhden.

[9] Tietyyppikohtainen, pätee koko tielle.

Kohteen sijainti alueella

R20: K_i or K_p / K_l or max(min(K_d / K_r,1),0)

Kohde kokonaan alueen sisällä (boolean)	K_i
Kohteen suurin alueen sisällä oleva pituus/säde	K_p
Kohteen kokonaispituus/säde	K_l
Kohteen etäisyys lähimmästä alueen rajaviivasta	K_d
Alueen säde[10]	K_r

Kahden kohteen sijainti samalla alueella, alue tunnettu[11]

R21a: $R20_1$ and $R20_2$

Kohde x sijaitsee alueella	$R20_x$

Kahden kohteen sijainti samalla alueella, alue tuntematon[12]

R21b: $K_{t1} = K_{t2}$ and $MAX_{path}(K_t = K_{t1}$ and $K_t = K_{t2})$

Maaston tyyppi	K_t
Parhaiten maastotyypiltään kohdepisteitä vastaava polku	MAX_{path}

Kahden kohteen sijainti samalla linjalla[13]

R22: K_l or $AND_{path}(K_{li})$ or ($R18_1$ and $R18_2$ and $AND_{path}(K_{li})$)

Looginen kytkentä kohteiden välillä (boolean)	K_l
Looginen kytkentä polun kahde peräkkäisen kohteen välillä (boolean)	K_{li}
Kohde x on lähellä polun osaa	$R18_x$

Kahden kohteen lähekkäisyys

R23: R18 or R19 or R22

Kohde on ympäristötekijän vaikutusalueella (saasteet, kaatopaikka, kosteus)

R24: $OR_p(K_d < K_r)$ or $OR_a(R20_a)$

Etäisyys pistemäisen ympäristötekijän lähteestä	K_d
Pistemäisen ympäristötekijän vaikutussäde	K_r
Sijainti ympäristötekijän alueella	$R20_a$
Kaikkien pistemäisten ympäristötekijöiden	OR_p

[10] Vapaamuotoisille alueille tämä on laskettavissa mutta tulos on hieman epämääräinen arvio. Tulokset voivat silti olla mielekkäitä.

[11] Tyypillisesti jokin hallinnollinen alue. Alueella on oltava viivoilla määritellyt rajat.

[12] Tyypillisesti jokin maastoalue, esim. sama suo, saman järven ranta. Alue ei voi olla kovin iso, koska säännössä joudutaan tekemään laajempi polunhaku kuin läheisyysarviossa. Järjestelmässä on oltava jokin maksimi polun pituudelle.

[13] Esimerkiksi sähkölinja. Sääntö tarkoittaa että kohteet ovat samalla linjalla jos niiden välillä on verkon särmä tai jos ne molemmat ovat lähellä samaa verkon särmää.

yhteisvaikutus
Kaikkien alueellisten ympäristötekijöiden yhteisvaikutus OR_a

Kohteen ympäristö on tietyn tyyppinen

R25: $K_t = K_{to}$ or $OR_{ca}(K_t = K_{ti}$ and $R20_{ca})$
Kohteen maasto K_{to}
Tutkittava maastotyyppi K_t
Lähiympäristön ruutujen maastotyyppi K_{ti}
Kohteen sijainti lähiruutujen lähellä $R20_{ca}$
Läheisten alueruutujen yhteisvaikutus OR_{ca}

7.4.2 Verkostosäännöt

Kohde on solmupiste

R26: $K_o = 1$ or $K_o > 2$ or $(K_o = 2$ and $K_n > 1)$
Pisteestä ulos lähtevien särmäviivojen lukumäärä K_o
Pisteessä olevien pistemäisten laitteiden lukumäärä K_n

Kahden vierekkäisen solmun välinen särmä on lyhyt

R27: $K_l = MIN_{ap1}(K_{le})$ and $K_l = MIN_{ap2}(K_{lp})$ and $K_l < K_{lmax}$
Solmujen välisen särmän pituus K_l
Kaikkien särmien pituus K_{le}
Pienin solmusta x lähtevän särmän pituus MIN_{apx}
Lyhyen maksimipituus K_{lmax}

Paras reitti kahden solmupisteen välillä

R28: $\Sigma_{path}(K_l) = MIN_{ap12}(K_{lp})$ and not$(MAX_{path}(K_u))$
Kahden solmun välisen särmän pituus K_l
Kaikkien polkujen pituus K_{lp}
Pienin solmun 1 ja 2 välisen polun pituus MIN_{ap12}
Särmän käyttöaste K_u
Solmun 1 ja 2 välisen polun suurin käyttöaste $MAX_{path}(K_u)$

Verkon polku on kriittinen[14]

[14] Kriittinen verkonosa tarkoittaa yleensä runkoverkkoa tai runkoverkkoliittymää. Se voi olla myös eri jakeluverkon osia yhdistävä polku jonka päässä on paljon tilaajia. Säännön tuloksellinen käyttö edellyttää että sitä on käytetty kaikille alaverkoille jotka tarkasteltavaan verkon osaan liittyvät. Kriittisyys on talletettava solmupistekohtaisesti tai särmäohtaisesti. Säännön soveltaminen on tehtävä depth-first-hakuna jossa otetaan huomioon silmukat.

R29: $\Sigma_{path}(K_s) > K_n$ or $MAX_{path}(K_{cs})$ or $\Sigma_{path}(K_c) > K_{nc}$

Polun solmusta lähtevien aliverkkojen lukumäärä	K_s
Polun solmujen lukumäärä	K_n
Alipolun kriittisyys	K_{cs}
Solmun polkujen päässä olevien asiakkaiden lukumäärä	K_c
Iso asiakkaiden määrä	K_{nc}

Polku on vakaa

R30: not($MAX_{path}(K_{te})$)

Maaston uhka kyseiselle särmätyypille[15]	K_{te}

Solmupiste on varmistettu

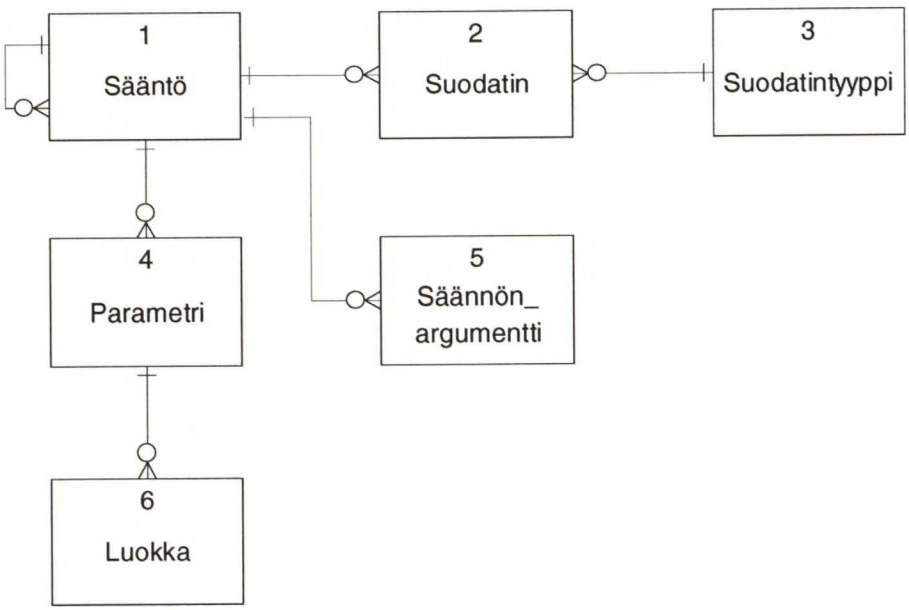
R31: $K_{np} > 2$ or R30

Vaihtoehtoisten polkujen lukumäärä	K_{ns}

[15] Esimerkiksi maakaapeli on vakaampi kuin ilmakaapeli. Uhkaava maastotekijä voi olla esimerkiksi ilmakaapelin lähellä olevat puut tai runsas linnusto, tai maakaapelille usein tehtävät kaivuutyöt kaupunkialueella.

8 Paikkatiedonhallinnan sääntöjen toteutus

Paikkatiedonhallinnan säännöt toteutetaan samoilla sumean logiikan rakenteilla kuin muutkin säännöt. Toiminnallisia laajennuksia tarvitaan kuitenkin jonkin verran. Samoin liittymiä muihin systeemeihin tarvitaan huomattavasti enemmän: esimerkiksi tieverkosto- ja maastomallit. Tässä kappaleessa on hahmoteltu paikkatiedonhallinnan vaatimia muutoksia alkuperäiseen sumean logiikan toteutukseen. Säännöt vaativat suuren määrän hakuja ja ovat varmasti erittäin raskaita, joten lopullinen toteutus vaatii ennenkaikkea optimointia.



Uusi ER-kaavio sumean logiikan käsittelyyn.

Säännön_argumentti:

- Argumentti_id
- Sääntö_id
- Tyyppi
- Tiedosto

Säännön argumentti on olemassa ainoastaan jos niitä tarvitaan. Ne on olemassa kaikille tarvittaville argumenteille jos yksikin tarvitsee argumentissä määriteltyjä lisätietoja. Tyyppi määrittelee minkä tyyppinen tieto vaaditaan, esim. POINT, LINE, AREA, ANY, NONE. Esim. R20: kohteen sijaniti alueella: toisen argumentin on oltava alue. Tiedosto määrittelee mistä luetaan järjestelmän ulkopuolinen määrittely: esimerkiksi tiestö tai maastomalli. Tällainen argumentti on implisiittinen; tyypiksi tulee NONE.

Hieman muutoksia toiminnallisuudessa tarvitaan myös sen vuoksi että useimmat sijaintitietoihin liittyvät säännöt liittyvät kahteen kohteeseen, joskus jopa kolmeen.

Sumean logiikan aiemmin esitelty malli käsittelee joko yhtä tai kaikkia kohteita kerrallaan. Näiden sääntöjen yhdistäminen tavanomaisiin sääntöihin tapahtuu s.e. kaikki muut säännöt lasketaan ainoastaan ensimmäiselle kohteelle, mutta paikkatiedonhallintaan liittyvät säännöt lasketaan molemmille. Yhdistäminen tapahtuu siis edelleen datan kautta, mutta käyttävässä sovelluksessa on otettava huomioon se että toinen tarkasteltavista kohteista on laskettava aiemmin, ilman paikkatiedonhallinnan sääntöjä. Näin on esimerkiksi kunnossapidossa: optimointivaiheessa on tarkasteltava kohde jonka kunto on jo laskettu ja lisäksi etsitään sen kanssa yhteen kuuluvia kohteita kaikista muista, jotka voisivat olla myös aiemmin laskemattomia. Kunnossapidossa vaiheittain etenemisen vuoksi niillekin on jo laskettu kuntoarvio ja laadittu alustava aikataulu.

Sijaintitietosäännöt toteutetaan sääntö-taulun uutena operaationa, lisänä and- ja or-operaatioihin, silloinkin kun käsitellään yhtä tai kolmea kohdetta. Kohteet ovat usein eri tyyppisiä: esimerkiksi aluekäsittelysäännöt vaativat viimeiseksi parametriksi alueen, muut voivat olla mitä graafisia kohteita tahansa. Sisäisen toteutuksen on otettava huomioon graafisten kohteiden kolme perustyyppiä, jollei sääntö erityisesti rajaa syötettä johonkin tiettyyn tyyppiin. Esim. R20: kohteen sijainti alueella: piste, viiva tai alue voi sijaita alueella, mutta mikään ei voi sijaita pisteessä tai viivassa. Niinpä kohteet rajataan s.e. 1. argumentti voi olla mikä vain, mutta 2. voi olla ainoastaan alue.

Varsinaisen sovelluskoodin puolella joudutaan tarjoamaan erilaisia vastakutsu-funktioita (callback-funktioita) joilla saadaan erityisesti verkon käsittelyyn tarvittavia sovelluskohtaisia palveluita toteutettua muuttamatta sumean logiikan käsittelyä sovelluskohtaisesti. Näitä callback-funktioita tarvitaan mm. ympäristekijöiden aiheuttavien kohteiden hakuun, verkon särmien ja solmujen hakuun (solmut särmästä, särmät solmusta, polun haku kahden solmun välistä) ja särmän priorisointi.

Algoritmi on kuvattu liitteessä 2. Algoritmi määrittelee ainoastaan sumean järjestelmän osuuden paikkatiedonhallinnasta. Verkon reitinhakuongelman ratkaisun kuvaus ei kuulu tähän työhön, varsinkin koska sellaiset palvelut ovat jo käytettävissä. Tämän työn puitteissa siitä käsitellään ainoastaan jollain tapaa sumeaan järjestelmään liittyviä osia, koko algoritmi on huomattavasti laajempi.

9 Yhteenveto

9.1 Työn tulokset ja niiden arviointi

Työssä on saatu selvitettyä sumean logiikan yleisiä soveltamismahdollisuuksia tietojärjestelmissä ja on varmistettu että on mahdollista hyödyntää sumeaa logiikkaa kunnossapitosovelluksessa. Varsinainen toteutus vaatii kuitenkin tarkemman suunnittelun, tämä työ on vasta todennut että on mahdollista toteuttaa sumeaa logiikkaa hyödyntävä kunnossapitosovellus. Suorituskyky ja käytettävyys ovat vielä epäselviä koska käytännön toteutusta ei vielä ole olemassa. Koska nykyisen järjestelmän suorituskyky on riittävä, ja koska sumean logiikan käsittely lisää enimmäkseen laskentaa, ei tietokantahakuja muistissa olevan datan ulkopuolelle, suorituskyky on luultavasti optimoitavissa riittäväksi.

Paikkatiedonhallinnan sumeat säännöt osoittavat että sumeaa logiikkaa voi soveltaa myös sijaintipohjaisiin riippuvuuksiin. Toteutettavuustutkimus on tehtävä vielä erikseen; säännöt edellyttävät melko monimutkaisia hakuja eikä suorituskyvyn riittävyydestä ole takeita.

Suurin ongelma tämän työn kannalta lienee saatavilla olevan materiaalin vähäisyys. Sumeasta logiikasta löytyy matemaattista ja automaatiokirjallisuutta riittävästi mutta niistä ei ole juuri apua tämäntyyppisessä sovelluksessa. Sama ongelma oli myös kunnossapidon sääntöjen kehittämisessä: sähköverkkojen kunnossapito on täysin laitoskohtaisten käytäntöjen varassa, siitä ei löydy juuri mitään kirjallista mateeriaalia. Materiaalin vähyyden vuoksi huomattavan suuri osa tässä työssä olevasta aineistosta on selvitetty itse, ei haettu kirjallisuudesta: kaikki kunnossapidon ja paikka- ja verkkotiedonhallinnan säännöt, samoin toteutus. Tämän vuoksi tulosten luotettavuus on kyseenalainen. Ne eivät varmasti ole täysin oikeita mutta niiden pitäisi olla oikeansuuntaisia ja niiden jatkokehittäminen tältä pohjalta pitäisi olla helpompaa.

9.1.1 Menetelmät

Työn tekeminen aloitettiin kirjallisuustutkimuksella jonka tuloksena oli että kirjallisuutta ei juurikaan ole. Tietoverkoista oli saatavana jonkun verran tietoa sumean logiikan soveltamisesta mutta sitä vaivasi sama ongelma kuin kirjallisuuttakin: enimmäkseen automaatiosta. Sumean logiikan matemaattisiin perusteisiin tällä tavoin pystyi kuitenkin perehtymään, mikä oli ensiarvoisen tärkeää sääntöjen kehittämisen kannalta.

Kunnossapidon järjestelmä pohjautuu suurelta osin haastatteluihin joilla pyrittiin kartoittamaan sähkönjakelulaitosten nykyisiä käytäntöjä. Käytännöt olivat enimmäkseen mittaustuloksin ja raja-arvoihin perustuvia, ja vahvasti käytännön kokemukseen ja intuitioon nojaavia. Niistä saatiin kuitenkin suunnitellulla kyselyllä kartoitettua olennaiset kohteet joita on käsiteltävä ja joukko pohjasääntöjä joista pystyi jatkamaan lopputulokseen.

Paikkatiedonhallinnan säännöt ja toteutus ovat puhtaasti omaan kokemustietoon ja kehitykseen pohjautuvia. Mikäli vastaavaa kehitystyötä on muualla tehty, tuloksia ei ole julkaistu laajemmalti.

9.2 Jatkokehitys

9.2.1 Toteutus

Kunnossapidon sumea järjestelmä on todettu mahdolliseksi toteuttaa. Sen suorituskyvyn ja hyödynnettävyyden arviointi on mahdollista vasta kun siitä on olemassa jonkinlainen toimiva versio. Samoin yleisempää hyödynnettävyyttä voidaan arvioida paremmin kun on olemassa jotain toimivaa. Lopullinen tuotteistaminen vaatii jo enemmän panostusta. Tällöin on myös toteutusta optimoitava jotta käytännön suorituskyky saataisiin riittäväksi. Esimerkiksi mahdollisia optimointeja ovat sääntöjen lukeminen keskusmuistiin ja sääntöjen suorituksen keskeyttäminen heti kun lopputulos on selvillä.

9.2.2 Aritmeettisten operaatioiden lisäys parametreihin, hierarkiakäsittely

Parametreilla ei pysty vielä mallittamaan kaikkia sääntöjä. Olennaisin lisäpiirre on aritmetiikan käsittely. Yksitasoinen hierarkia ei myöskään ole riittävä, jos aritmetiikkaa käsitellään tasoja on voitava olla rajattoman monta. Tämä lienee hyödyllinen ominaisuus muuallakin. Rajanveto sumeiden totuusarvojen ja laskutuloksien välillä voi aiheuttaa ongelmia.

9.2.3 Lähtöarvojen virheiden käsittely

Tämä jatkokehitystarve lähtee suoraan kunnossapitosovelluksen tarpeista: mittaustuloksilla on merkittävä osuus sääntöjen tuottamista lopputuloksista. Järjestelmä ei kuitenkaan ota millään tavalla huomioon mittaustuloksien luotettavuutta. Erityisesti mikäli järjestelmään lisätään perustelu-ominaisuuksia on lähtöarvojen luotettavuus saatava otetuksi huomioon.

Lähtöarvot voivat olla epäluotettavia lukemattomista eri syistä. Mittaustuloksissa tämä on selvää, mutta kaikki ihmisten syöttämät ja/tai hankkimat tiedot ovat jossain määrin epäluotettavia. Kaikkia lähtöarvoja ei voi luokitella epäluotettaviksi, muutoin mielekkäiden tuloksien saaminen muuttuu mahdottomaksi. Lähtöarvoja voidaan kuitenkin pitää epäluotettavina seuraavissa tapauksissa:

- Arvo poikkeaa voimakkaasti (esim. dekadilla) muista vastaavista arvoista, ilman ilmeistä yhteyttä mihinkään tunnettuun tekijään. Mikäli kaikki kohteen arvot poikkeavat, on todennäköisempää että tulokset ovat oikein, kuin jos ainoastaan yksi tulos on radikaalisti erilainen kuin yleensä.
- Täysin raja-arvojen ulkopuolella olevat arvot ovat tietenkin aina kelvottomia (esim. negatiivinen resistanssi).
- Mikäli parametrissa oleva painoarvo on alle 1:n, sen luotettavuus on vastaavassa määrin huonompi ja sen saama painoarvo lopputuloksessa pienempi.

9.2.4 Absoluuttiset rajoitteet yleisesti

Kunnossapitosovelluksessa olisi hyödyllistä voida hallita kustannuksia, työvoimaa ja muita resursseja. Tämänkaltaiset ei-sumeat rajoitteet on tehtävissä yleisesti sumean järjestelmän yhteyteen käsittelemällä niitä normaalin Boolen logiikan säännöillä. Sumean järjestelmän osalta tämä vaatii totuusarvojen tyypittämistä joko sumeiksi tai ei-sumeiksi, ja Boolen and-, or- ja not-operaatioiden lisäämistä muiden joukkoon.

9.2.5 Perustelut

Käyttäjälle on ensiarvoisen tärkeää että tietojärjestelmän hänellä antamat tiedot ovat oikeita. Silloin kun tästä ei voida olla varmoja, on lähes yhtä tärkeää että käyttäjä saa tietää kuinka luotettava hänen saamansa tieto tai toimenpide-ehdotus on. Erityisesti silloin kun luotettavuus vaihtelee voimakkaasti, kuten esimerkiksi tässä dokumentissa määritellyn kunnossapitojärjestelmän tapauksessa on, luotettavuusarvio ja tieto siitä kuinka tulokseen on päädytty on erittäin hyödyllinen.

Lopputuloksen luotettavuus riippuu syötteen luotettavuudesta ja kaikkien välivaiheiden yhteisvaikutuksen luotettavuutta laskevasta vaikutuksesta. Käyttäjälle näytettävät perustelut koostuvat pääasiassa kahdesta tiedosta:

1. Luotettavuusarvio lukuarvona, joka saadaan laskennallisesti sääntöjen yhteisvaikutuksesta. Mikäli tulos on saatu suoraan ehdottoman luotettavasta lähtöarvosta, on luotettavuusindeksi 1. Kaikissa muissa tapauksissa se on pienin olennaisesti tulokseen vaikuttaneiden tekijöiden luotettavuuksista vähennettynä välivaiheiden aiheuttamalla kumulatiivisella luotettavuuden heikkenemisellä. Tämä on pitkälti analoginen normaalien fysikaalisten laskutoimitusten virhemarginaalien laskennan kanssa. Luotettavuusarvio on eräänlainen worst-case-analyysi: arvo ilmaisee kuinka luotettava lopputulos ainakin on.
2. Perustelut jossa luetellaan kaikki tulokseen vaikuttaneet lähtöarvot luotettavuusarvoineen sekä sääntöketju jolla lopputulokseen on tultu. Kaikki säännöt jotka ovat lauenneet ja vaikuttaneet lopputulokseen näytetään, mutta erityisesti korostaen sitä sääntöketjua joka tuotti huonoimman luotettavuusarvion. Laskennallinen luotettavuusarvo laskee yleensä niin voimakkaasti jo parin operaation jälkeen. Käyttäjä voi hyväksyä hyvinkin alhaisen luotettavuusarvion saaneen tuloksen, tarkastettuaan kuinka siihen on päädytty.

9.2.6 Neuraaliverkot

Säännöt jotka tässä työssä on määritelty eivät ole missään määrin lopullisia. Niitä tullaan täydentämään, muuttamaan ja hienosäätämään kunnes ne vähitellen saavat haluttuja tuloksia aikaan kaikilla syötteillä. Tämänkaltainen iteraatio vie kuitenkin runsaasti aikaa ja resursseja. Mikäli pystytään antamaan selkeät ohjeet tapaus tapaukselta minkälainen tulos kutakin syötettä vastaa, voidaan tätä opetusaikaa lyhentää käyttämällä neuraaliverkkoja sääntöjen hienosäätöön. Uusien, täysin riippumattomien tekijöiden lisääminen näin ei onnistu mutta muutoin tulokset ovat usein erittäin hyviä.

Neuraaliverkkoa voidaan soveltaa myös alkuopetuksen jälkeen sääntöjen optimointiin kertomalla sille jatkuvasti milloin sen tekemiä ehdotuksia ei ole hyväksytty, jolloin se säätää sääntöjä koko ajan lähemmäs haluttua. Jossain vaiheessa tulee tietenkin raja vastaan, jonka jälkeen neuraaliverkko ei enää saa parannettua lopputulosta vaan se lähinnä jää värähtelemään lähelle löydettyä optimipistettä.

Neuraaliverkot voivat myös parantaa lopputulosten luotettavuutta. Suoranaisten luotettavuuslaskelmien lisäksi voi neuraaliverkosta tulla vaikutusta luotettavuusarvioon

sen aiempien kokemuksien perusteella. Tämä edellyttää optimoinnin lailla neuraaliverkolle jatkuvasti syötettävää tietoa onko sen tekemät päätökset olleet oikeita vai vääriä.

Neuraaliverkoille on mahdollista opettaa muillakin tavoilla ja niillä on lukuisia muitakin käyttökohteita kuin edellämainitut. Niitä ei kuitenkaan ole tämän pidemmälle tämän työn puitteissa selvitetty.

Neuraaliverkkojen käyttöönotossa on ongelmana niiden heikohko soveltuvuus tietojärjestelmiin. Ne ovat tavanomaisella tietokonelaitteistolla hitaita ja paljon muistia vaativia. Yleensä niitä käytetään erikoistuneella neuraaliprosessoreilla joissa on tuhansia yksinkertaisia prosessoreita jotka kykenevät tämäntapaiseen laskentaan paremmin. Neuraaliverkkojen toteutuksesta tietojärjestelmiin ei ole olemassa juurikaan kirjallisuutta, joten toteutus vaatii vähintään saman verran tutkimusta ja suunnittelua kuin tässä dokumentissa esitellyn sumean päättelyn koneistonkin suunnittelu. Joka tapauksessa käytännön kapasiteettirajoitukset estävät kovin monimutkaisten rakenteiden soveltamisen; erikoistunutta laitteistoa ei tietojärjestelmään saa integroitua.

9.2.7 Kausaaliverkot

Usein sumeat joukot joista säännöt muodostuvat eivät ole riippumattomia vaan kahden tai useamman tekijän välillä on jokin yhdistävä tekijä. Ei-triviaaleissa tapauksissa tämä johtaa helposti joko vääriin tuloksiin tai hankaliin rakenteisiin joilla nämä riippuvuudet saadaan kuvattua. Kausaaliverkot ovat eräs tapa kuvata tämänkaltaisia riippuvuuksia. Niitä ei ole tämän työn yhteydessä tutkittu, koska ne kausaaliverkkojen teoria on matemaattisesti varsin hankalaa. VTT:n tutkimusprojektissa on kuitenkin tarkoitus tuottaa välineistöä tällaisten verkkojen mallittamiseen. Projektin edistyessä voidaan näiden hyödyntämistä tutkia tarkemmin.

9.2.8 Soveltaminen muuntyyppisiin järjestelmiin

Paikkatiedonhallintajärjestelmiä on monentyyppisiä, joista verkkotietojärjestelmät ovat yksi erikoistapaus. Joissakin alueiden hallinta on yksi pääkäyttötarkoituksista, joissakin olennaista on ainoastaan pistemäisten kohteiden sijainnit. Aiemmin kuvattu sumean logiikan toteutus sopii yhtä hyvin näihin kaikkiin, myös paikkatiedonhallinnan sääntöjen osalta. Sovellukset vain painottuvat eri sääntöihin ja mekanismeihin. Toteutuksen määrittely ei kuitenkaan ole täydellinen edes nykyiseen käyttöön, varsinaista toteutusta varten on määriteltävä huomattavasti tarkemmin rajapinnat ja sisäinen toteutus jotta riittävä yleiskäyttöisyys voidaan saavuttaa. Lisäksi on vielä melko paljon sellaisia määriteltyjä toiminnalisuuksia jotka voisi saada eristettyä dataksi ohjelmakoodin sijasta. Tämä on tarpeen koska monimutkaiset säännöt eivät saa olla tiukasti kiinni ohjelmakoodissa jos se suinkin on vältettävissä.

Aiemmin kuvatut paikkatiedonhallinnan säännöt eivät ota huomioon kuin kaksiulotteisen tilan. Esimerkiksi rakennusten kuvauksessa on kuitenkin ehdottoman välttämätöntä saada myös kolmas ulottuvuus hallittua. Tämä monimutkaistaa sääntöjä jonkin verran, mutta on kuitenkin mahdollista toteuttaa. Aluekäsitteet on muutettava kolmiulotteiseksi, ja verkostojen särmien määrittely muuttuu myös. Sen sijaan esim. verkostojen solmut

käsitellään edelleen täysin samalla tavalla, mahdolliset erot saadaan siirrettyä sovelluksen puolelle callback-funktioiden avulla.

Hyödyllinen piirre voisi olla myös ajan huomioonottaminen säännöissä. Tämä vaatii huomattavasti enemmän tutkimista, mutta monet reaalimaailman ilmiöt ja sanonnat liittyvät aika-käsitteeeseen joten tarve on ilmeinen. Esimerkiksi voitaisiin haluta ilmaista matkan pituutta suhteessa käytettävissä olevaan aikaan. Ilmeisesti kuitenkin ajan käsittelyn säännöt ovat täysin erillisiä kaikesta mitä tässä dokumentissa on käsitelty. Tarpeiden kartoitus olisi ensimmäinen askel tällaisten sääntöjen kehittämiseen.

10 Lähdeluettelo

1 Sähkönjakeluyhtiöiden ja IVOn henkilökunnan haastattelut
2 Jarmo Elovaara-Yrjö Laiho 1988 Sähkölaitostekniikan perusteet
3 Jorma Mörsky-Janne Mörsky 1994 Voimalaitosten yhteiskäytön tekniikka
4 Earl Cox 1994 The Fuzzy Systems Handbook
5 Bart Kosko 1992 Neural Networks and Fuzzy Systems
6 General Electrics seminaari 1995 Piero Bonissone-Pratap Khedkar
A Workshop for Finnish Industry in Fuzzy Logic and Neural Networks

Liite 1: Ohjelmaesimerkki

1.1 Periaatteet, rakenne ja nimeämiskäytäntö

Ohjelmaesimerkit pohjautuvat toteutus-luvussa esitetyille periaatteille. Kaikki esimerkit on tehty C-kielellä. Koska kokonaisrakenne on kuvattu siellä, kuvataan tässä luvussa ainoastaan sumean logiikan käsittelyn perusrunko kooditasolla ja keskitytään enemmän käyttöesimerkkeihin. Luvussa esitetään myös esimerkkidataa riittävästi jotta lukija pystyy saamaan kuvan datan rakenteesta ja järjestelmän toiminnasta sen pohjalta.

Funktiot nimetään kolmikirjaimisella modulitunnisteella sekä kuvaavalla nimellä yhteen kirjoitettuna jonka kaikki sanat alkavat isolla kirjaimella. Muuttujanimet muodostetaan muutoin samoin mutta ilman modulietuliitettä. Makrot ja esiprosessorimäärittelyt kirjoitetaan kokonaan isoilla kirjaimilla.

Tietokantakäsittely oletetaan tapahtuvaksi valmiin rajapinnan läpi (modulitunniste rdb). Tietokantatauluilla on määritelty niitä vastaavat numerot joita käytetään tietokantapalveluiden ohjaamiseen. Eri kysely- ja päivitysoperaatiot erotetaan nimillä jotka annetaan ym. tietokantapalveluille parametreina. Kullekin tietokantataululle on määritelty C-struktuuri joka on yhdenmukainen sen kantamäärittelyn kanssa ja voi sisältää tietokannan rivin tiedot. Nämä struktuurit ovat r_tietokantataulu_t-tyyppisiä ja vastaavat muuttujanimet ovat mallia TietoKantaTaulu.

Käyttöesimerkeissä oletetaan koko sumean logiikan käsittelyosa olemassaolevaksi, vaikka sitä ei määritellä tässä työssä perusrunkoa lukuunottamatta. Mikäli esimerkissä käytetään perusrunkoon kuulumatonta palvelua, jonka nimi ei selitä sen toimintaa riittävästi, dokumentoidaan sen toimintaa kommentilla esimerkissä. Kuten tietokantataulujen rakenteessakin, on näistä esimerkeistä jätetty pois kaikki ylimääräiset osat ja rakennetta on yksinkertaistettu niin että siitä näkyy perusperiaatteet mutta se ei ole sellaisenaan toteuttamiskelpoinen.

1.2 Sumean logiikan toteutus

Rajapinta:

```
typedef  struct
{
    Table_e Table;
    void *Argument;
} Argument_t;

Success_e fzyCallRule(int RuleId, double *Result,
                      Argument_t *aArguments, int nArguments);
Success_e fzySetCallBack(int CallBackNumber, int(*)());


Success_e fzyCallRule(int RuleId, double *Result,
                      Argument_t *aArguments, int nArguments);
{
```

```
r_rule_t Rule, SubRule;
r_rule_argument_t RuleArgument;
r_parameter_t Parameter;
r_hedge_t Hedge;
RdbCursor_t Cur;
double Value = NaN;

*Result = 0;
Rule.id = RuleId;
if (rdbSelect(RULE, &Rule))
{
      ERROR("no such rule");
}
RuleArgument.idrule = RuleId;
Cur = rdbOpenCursor(RULE_ARGUMENT, "idrule",
                    &RuleArgument, &RuleArgument);
while (!rdbFetch(Cur, &RuleArgument))
{
      if (RuleArgument.type)
      {
            /* check type; return if fails. */
      }
      if (!genIsEmptyString(RuleArgument.file))
      {
            /* open and store file pointer; return if
            fails. */
      }
}
SubRule.idrule = RuleId;
Cur = rdbOpenCursor(RULE, "idrule",
                        &SubRule, &SubRule);
while (!rdbFetch(Cur, &SubRule))
{
      if (fzyCallRule(SubRule.id,
                        &Value, aArguments, nArguments))
      {
            ERROR("alisääntö epäonnistui");
      }
      fzyApplyOperation(Rule.operation, *Result,
                        Value, Result);
}
Parameter.idrule = RuleId;
Cur = rdbOpenCursor(PARAMETER, "idrule_ordered",
                        &Parameter, &Parameter);
while (!rdbFetch(Cur, &Parameter))
{


      if (fzyCalculateParameter(&Parameter, &Value,
                                aArguments, nArguments))
      {
            ERROR("Parametrin laskenta epäonnistui.");
      }
      fzyApplyOperation(Rule.operation, *Result,
                        Value, Result);
}
if (*Result < Rule.alfacut ||
    (!Rule.alfacut && *Result < fzyGlobalAlfaCut()))
{
      *Result = 0.0;
}
Hedge.idrule = RuleId;
Cur = rdbOpenCursor(HEDGE, "idrule_ordered",
                    &Hedge,&Hedge);
```

```
    while (!rdbFetch(Cur, &Hedge))
    {
          if (fzyCalculateHedge(&Hedge, Result,
                                aArguments, nArguments))
          {
                ERROR("Suodattimen laskenta epäonnistui");
          }
    }
    return(OK);
}

Success_e fzyCalculateParameter(r_parameter_t *Parameter,
                                double *Result,
                                Argument_t *aArguments,int nArguments)
{
    double Value;

    switch (Parameter->type)
    {
    case VALUE:
/* Find the right argument, find the offset of the field in Object
    pointer, cast it right type and return in value. Keeps track of
    used args in case there are  multiple of the same type.*/
          if (fzyFindFieldFromArgs(Parameter->table,
                                   Parameter->field,
                                   aArguments, nArguments, &Value))
          {
                ERROR("Argumentti puuttuu");
          }
          if (fzyCompareValue(Value, Parameter->lowerbound,
                              Parameter->upperbound,
                              Parameter->pointvalue, Result))
          {
                ERROR("Vertailu epäonnistui");
          }
          break;
    case CLASS:
          if (fzyFindFieldFromArgs(Parameter->table,
                                   Parameter->field,
                                   aArguments, nArguments, &Value))
          {
                ERROR("Argumentti puuttuu");
          }
          if (fzyFindClassValue(Value, Parameter->id, EXACT,
              Result))
          {
                ERROR("Luokam haku epäonnistui.");
          }
          break;
    case CONTCLASS:
          if (fzyFindFieldFromArgs(Parameter->table,
                                   Parameter->field,
                                   aArguments, nArguments, &Value))
          {
                ERROR("Argumentti puuttuu");
          }
          if (fzyFindClassValue(Value, Parameter->id, CONTINUOUS,
              Result))
          {
                ERROR("Luokam haku epäonnistui.");
          }
          break;
    case COUNT:
    /* etc */
```

```
    case SUM:
    /*etc*/
    case MEAN:
    /*etc*/
    case LESS:
    /*etc*/
    /*etc*/
    }
    return(OK);
}
Success_e fzyCalculateHedge(r_hedge_t *Hedge, double *Result,
                            Argument_t *aArguments, int nArguments)
{
    r_hedgetype_t HedgeType;

    HedgeType.id = Hedge->idhedgetype;
    if (rdbSelect(HEDGETYPE, &HedgeType))
    {
          ERROR("Suodatintyyppiä ei löydy.");
    }
    switch (HedgeType.type)
    {
    case ZADEH_NOT:
          *Result = 1 - *Result;
          break;
    case YAGER_NOT:
    /* etc */
    default:
          if (HedgeType.effect >= 100)
          {
                *Result = (*Result < 0.5) ? 0.0 : 1.0;
                break;
          }
          else if (HedgeType.effect <= -100)
          {
                break;
          }
          *Result = pow(*Result, 100/(100-HedgeType.effect));
          break;
    }
    return(OK);
}
```

1.3 Sumean logiikan käyttö

Tietokannan esimerkkisääntö:

Sääntö

Id	1
Isäsääntö	0
Operaatio	YAGER_AND
Nollataso	0,2
Kuvaus	R8: Laitteen huollon tarp.
Painoarvo	1.0

Id	2
Isäsääntö	1
Operaatio	ZADEH_AND
Nollataso	0,2
Kuvaus	R3: Kuntoarvio, yhdistelmä
Painoarvo	1.0

Id	3
Isäsääntö	2
Operaatio	ZADEH_AND
Nollataso	0,2
Kuvaus	R2: Kuntoarvio, alilaiteet
Painoarvo	1.0

Id	4
Isäsääntö	2
Operaatio	ZADEH_AND
Nollataso	0,2
Kuvaus	R1: Kuntoarvio, parametrit
Painoarvo	1.0

Id	5
Isäsääntö	1
Operaatio	NONE
Nollataso	0,2
Kuvaus	R4: Määräaikaishuollon tarp
Painoarvo	1.0

Id	6
Isäsääntö	1
Operaatio	ZADEH_OR
Nollataso	0,2
Kuvaus	R5: Huollon taloudellisuus
Painoarvo	1.0

Id	7
Isäsääntö	1
Operaatio	ZADEH_AND
Nollataso	0,2
Kuvaus	R7: Huollon tekninen tarve
Painoarvo	1.0

Id	8
Isäsääntö	7
Operaatio	ZADEH_AND
Nollataso	0,2
Kuvaus	R6: Tyypin huoltotarve
Painoarvo	1.0

Parametri

Id	1
Kuvaus	Ikääntyminen
Sääntö_id	4
Parametri_id	0
Tyyppi	CONTCLASS
Painoarvo	1
Taulu	katkaisija
Kenttä	ikä
Alaraja	0.0
Ylaraja	0.0
Pistearvo	0.0
Kenttäarvo	0.0

Id	2
Kuvaus	Toiminta-aika auki nor.ohj.
Sääntö_id	4
Parametri_id	0
Tyyppi	VALUE
Painoarvo	1
Taulu	kp_tieto
Kenttä	arvo
Alaraja	0.0
Ylaraja	30.0
Pistearvo	0.0
Kenttäarvo	0.0

Id	3
Kuvaus	Toiminta-aika kiinni nor.ohj
Sääntö_id	4
Parametri_id	0
Tyyppi	VALUE
Painoarvo	1
Taulu	kp_tieto
Kenttä	arvo
Alaraja	0.0
Ylaraja	140.0
Pistearvo	0.0
Kenttäarvo	0.0

Id	4
Kuvaus	virityaika s
Sääntö_id	4
Parametri_id	0
Tyyppi	VALUE
Painoarvo	1
Taulu	kp_tieto
Kenttä	arvo
Alaraja	10.0
Ylaraja	15.0
Pistearvo	0.0
Kenttäarvo	0.0

Id	5
Kuvaus	toimintajännite (kela) V
Sääntö_id	4
Parametri_id	0

Tyyppi VALUE
Painoarvo 1
Taulu kp_tieto
Kenttä arvo
Alaraja 180.0
Ylaraja 0.0
Pistearvo 0.0
Kenttäarvo 0.0

Id 6
Kuvaus Ylimenovastus μΩ
Sääntö_id 4
Parametri_id 0
Tyyppi VALUE
Painoarvo 1
Taulu kp_tieto
Kenttä arvo
Alaraja 0.0
Ylaraja 200.0
Pistearvo 0.0
Kenttäarvo 0.0

Id 7
Kuvaus Toimintakertojen lkm
Sääntö_id 4
Parametri_id 0
Tyyppi CONTCLASS
Painoarvo 1
Taulu katkaisija
Kenttä käyttökerrat
Alaraja 0.0
Ylaraja 0.0
Pistearvo 0.0
Kenttäarvo 0.0

Luokka
Id 0
Parametri_id 1
Tunnus Uusi
Arvo 1.0

Id 30
Parametri_id 1
Tunnus Puoliväli
Arvo 0.65

Id 60
Parametri_id 1
Tunnus Käyttökelvoton
Arvo 0.0

Id 0
Parametri_id 7
Tunnus Käyttämätön
Arvo 1.0

Id 1000
Parametri_id 7
Tunnus Käyttökerrat täysi
Arvo 0.0

Suodatin
Sääntö_id 2
Järjestysnumero 1
Suodatintyyppi_id 1

Suodatintyyppi

Id 1
Kuvaus Ei
Tyyppi not
Vaikutus 0

1.4 Kunnossapidon toteutus

Esimerkki yksittäisen kohteen kuntotiedoista: katkaisija

Huoltoväli kk	24
Tarkastusväli kk	6
Elinikä	60

Mittaus	Alaraja	Yläraja	Muoto
Toiminta-aika auki norm.ohjaus ms	0	30	Yläkynnys
Toiminta-aika kiinni norm.ohjaus ms	0	140	Yläkynnys
Viritysaika s	10	15	Arvoväli
Toimintajännite (kela) V	180	0	Alakynnys
Ylimenovastus μΩ	0	200	Yläkynnys
Toimintakertojen lkm	0	1000	Lineaarinen, neg.
Ikä kk	0	60	Lineaarinen, neg.

Käyttävä ohjelmakoodi:

```
Success_e mntNeedsRepair(Table_e Table, void *Object,
                         double *Result)
{
   Argument_t *aArguments;
   int nArguments = 0, MaxArguments = 100;
   RdbCursor_t Cur;
   r_maintdata_t MaintData;

   aArguments =
       genTempMalloc(MaxArguments*sizeof(Argument_t));
   aArguments[nArguments].Object = Object;
   aArguments[nArguments++].Table = Table;
   MaintData.objectid = ID(Table,Object);
   Cur = rdbOpenCursor(MAINTDATA, "object_id",
                       &MaintData, &MaintData);
   while (!rdbFetch(Cur, &MaintData))
   {
        aArguments[nArguments].Object =
             genDupObject(MaintData);
        aArguments[nArguments++].Table = MAINTDATA;
        /* check bounds & realloc */
   }
   if (fzyCallRule(1, Result, aArguments, nArguments))
   {
        ERROR("Kuntoarvio epäonnistui");
        return(ERROR);
   }
   return(OK);
}
```

Suorituksen kulku:

Tarkasteltavan katkaisijan tiedot:

Mittaus	Mittaustulos	Arvo
Toiminta-aika auki norm.ohjaus ms	28	0.97
Toiminta-aika kiinni norm.ohjaus ms	142	0.8
Viritysaika s	13	1.0
Toimintajännite (kela) V	188	1.0
Ylimenovastus μΩ	179	1.0
Toimintakertojen lkm	653	1.0
Ikä kk	42	0.49

1.Haetaan pääsääntö eli sääntö numero 1 joka annettiin parametrina fzyCallRule-kutsussa.
2.Haetaan sääntöön 1 liittyvät parametrit; ei löydy.
3.Haetaan alisäännöt; löytyy säännöt 2, 5, 6 ja 7.
4.Haetaan sääntöön 2 liittyvät parametrit; ei löydy.
5.Haetaan alisäännöt; löytyy säännöt 3 ja 4.
6.Haetaan sääntöön 3 liittyvät parametrit; ei löydy.
7.Haetaan alisäännöt; ei löydy.
8.Haetaan sääntöön 3 liittyvät parametrit; löytyy 7 parametria.
9.Parametri 1 on tyyppiä CONTCLASS, haetaan sille luokkarajat; löytyy 3 luokkaa.
9.1.Haetaan annetuista tiedoista katkaisija-kohteen tieto ikä; saadaan tulos 42.
9.2.Haetaan löytyykö suoraan annetuista luokista; ei löydy.
9.3.Haetaan luokat 30 ja 60 joiden välissä arvo 42 on, ja lasketaan lineaarisesti sille arvo väliltä 0.65 ja 0.0; saadaan 0.65 - (42-30)/(60-30)*(0.65-0.0) = 0.49
10.Parametri 2 on tyyppiä VALUE.
10.1.Haetaan annetuista tiedoista kp_tieto-kohteen tieto arvo; saadaan mittaustulos 28.
10.2.Verrataan saatua arvoa parametrin raja-arvoihin; annettu yläraja-arvo joten verrataan siihen. Mittaustulos on hyvin lähellä ylärajaa joten laskennan tulos on 0.97.
11.Käydään läpi loput parametrit, laskien niille annetuista lähtötiedoista sumea totuusarvo.
12.Haetaan alisäännöt; ei löydy.
13.Yhdistetään parametrien ja alisääntöjen tulokset säännölle 4 käyttäen Zadeh:in and:iä, eli otetaan pienin saaduista arvoista lopputulokseksi. Tässä tapauksessa pienin on ikääntymisestä saatu 0.49.
14.Yhdistetään parametrien ja alisääntöjen tulokset säännölle 2 käyttäen Zadeh:in and:iä, eli otetaan pienin saaduista arvoista lopputulokseksi. Tässä tapauksessa pienin on säännöstä 4 saatu 0.49.
15.Sovelletaan suodatinta 1 säännön 2 lopputulokseen.
16.Haetaan sääntöön 5 liittyvät parametrit ja lasketaan niiden totuusarvo.
17.Haetaan alisäännöt; ei löydy.
18.Yhdistetään parametrien ja alisääntöjen tulokset säännölle 5. Koska operaatio on NONE, ei tuloksia pitäisi tulla kuin yksi.
19.Haetaan sääntöön 6 liittyvät parametrit ja lasketaan niiden totuusarvo.
20.Haetaan alisäännöt; ei löydy.
21.Yhdistetään parametrien ja alisääntöjen tulokset säännölle 6 käyttäen Zadeh:in or:ia.
22.Haetaan sääntöön 7 liittyvät parametrit ja lasketaan niiden totuusarvo.
23.Haetaan alisäännöt; löytyy sääntö 8.
24.Haetaan sääntöön 8 liittyvät parametrit ja lasketaan niiden totuusarvo.

25.Haetaan alisäännöt; ei löydy.
26.Yhdistetään parametrien ja alisääntöjen tulokset säännölle 8 käyttäen Zadeh:in and:iä.
27.Yhdistetään parametrien ja alisääntöjen tulokset säännölle 7 käyttäen Zadeh:in and:iä.
28.Yhdistetään parametrien ja alisääntöjen tulokset säännölle 1 käyttäen Yager:in and:iä.
29.Palautetaan lopputulos.

Parametrien laskennan jälkeen ja lopputulosten yhdistämisen yhteydessä jokaisen säännön tapauksessa tarkastetaan alittaako lopputulos nollatason. Nollataso on määritelty sekä yleisenä arvona että sääntökohtaisesti.

Aina säännön laskennan yhteydessä talletetaan tieto sen lopputuloksesta ja tietoja mistä sen tulos muodostuu.

Liite 2: Algoritmit kokonaisuudessaan

2.1 Sumean logiikan algoritmi

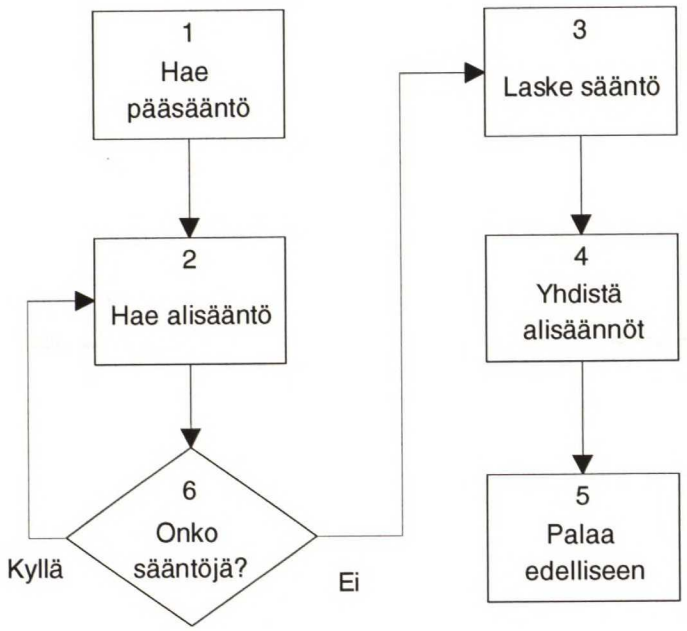
Sääntöhierarkian läpikäynti:

1 Haetaan pääsääntö sovelluksen antamilla kriteereillä. Tämä on liittymä varsinaiseen sovellukseen päin: syötteenä saadaan tieto mitä sääntöä sovelletaan ja mille kohteelle.

2 Haetaan säännön alasäännöt rekursiivisesti kunnes saavutetaan hierarkian lehtisäännöt. Hierarkia käydään läpi syvyys-ensin-hakuna.

3 Lehtisäännöistä alkaen lasketaan säännöille yksi kerrallaan arvot pohjautuen niiden parametreihin ja mahdollisien alasääntöjen arvoihin. Yksittäisen säännön arvon laskenta on kuvattu alla.

4 Yhdistetään alasääntöjen arvot.



Yksittäisen säännön laskenta:

1 Haetaan säännön parametrit. Sellaiset parametrit joille on annettu nollasta poikkeava parametri_id ohitetaan tässä. Niille on oma erityinen käsittelynsä hierarkisten parametrien tapauksessa.

2 Kullekin parametrille tutkitaan sen tyyppi:

2.1 Arvo: Haetaan tarkasteltavalle kohteelle taulu- ja kenttänimien perusteella lukuarvo argumenteista, ottaen järjestyksessä aina seuraava jos samantyyppisiä on useampia, ja verrataan sitä parametri-taulun raja-arvoihin:

2.1.1 Jos pistearvo on annettu, verrataan haettua arvoa siihen. Arvoksi tulee haetun arvon ja pistearvon läheisyys normaalijakaumalla.

2.1.2 Jos alaraja on annettu, verrataan haettua arvoa siihen: mikäli haettu arvo on selvästi suurempi, tulos on 1. Jollei rajaa ole annettu, arvo on 1.

2.1.3 Jos yläraja on annettu, verrataan haettua arvoa siihen: mikäli haettu arvo on selvästi pienempi, tulos on 1. Jollei rajaa ole annettu, arvo on 1.

2.1.4 Lopputulokseksi tulee pienin ym. arvoista.

2.2 Luokka: Haetaan tarkasteltavalle kohteelle taulu- ja kenttänimien perusteella luokka-arvo argumenteista, ottaen järjestyksessä aina seuraava jos samantyyppisiä on useampia.

2.2.1 Haetaan parametri_id:n ja haetun luokka-arvon perusteella luokka-taulusta vastaava arvo.

2.2.2 Mikäli luokkaa ei löydy, arvoksi tulee 0.

2.3 Jatkuva luokka: Haetaan tarkasteltavalle kohteelle taulu- ja kenttänimien perusteella arvo argumenteista, ottaen järjestyksessä aina seuraava jos samantyyppisiä on useampia.

2.3.1 Haetaan parametri_id:n perusteella luokka-taulusta vastaavat arvot. Mikäli joku vastaa tarkalleen haettua arvoa, lopputulos saadaan suoraan siitä.

2.3.2 Valitaan haetuista luokista ne kaksi joiden arvot ovat lähimpänä haettua arvoa sen eri puolilla (suurempi ja pienempi).

2.3.3 Mikäli suurempaa arvoa ei löydy lainkaan, lopputulos on 1. Mikäli pienempää arvoa ei löydy lainkaan, lopputulos on 0.

2.3.4 Lasketaan lopputulos suoralta näiden kahden arvon välistä: $A = A_s + (A_g - A_s) / (V_g - V_s) * (V - V_s)$

2.4 Lukumäärä: Tarkasteltavaa kohdetta ei anneta, vaan kaikki taulu-kentän määräämän taulun kohteet ovat tarkastelun kohteena. Argumenttia ei tarvita.

2.4.1 Haetaan taulu-kentän määräämästä taulusta sellaiset rivit joille parametrin määräämän kentän arvo on sama kuin parametrin kenttäarvo.

2.4.2 Lasketaan rivien lukumäärä.

2.4.3 Verrataan näin saatua lukumäärää samoin kuin arvo-tyypissä: pistearvo, ala- ja ylärajavertailuna.

2.5 Summa: Tarkasteltavaa kohdetta ei anneta, vaan kaikki taulu-kentän määräämän taulun kohteet ovat tarkastelun kohteena.

2.5.1 Haetaan taulu-kentän määräämästä taulusta kaikki rivit.

2.5.2 Lasketaan parametrin osoittaman kentän summa kaikista riveistä.

2.5.3 Verrataan näin saatua summaa samoin kuin arvo-tyypissä: pistearvo, ala- ja ylärajavertailuna.

2.6 Keskiarvo: Tarkasteltavaa kohdetta ei anneta, vaan kaikki taulu-kentän määräämän taulun kohteet ovat tarkastelun kohteena.

2.6.1 Haetaan taulu-kentän määräämästä taulusta kaikki rivit.

2.6.2 Lasketaan parametrin osoittaman kentän keskiarvo kaikista riveistä.

2.6.3 Verrataan näin saatua keskiarvoa samoin kuin arvo-tyypissä: pistearvo, ala- ja ylärajavertailuna.

2.7 Vertailuoperaattori: Vertailuoperaattoreilla on aina kaksi aliparametria. Niitä ei käytetä muuhun kuin hierarkisten parametrien vertailuun. Aliparametreille ei tehdä mitään yleensä oletusarvoisesti parametreille tehtäviä toimenpiteitä vaan tarkalleen ne mitä alla luetellaan.

2.7.1 Haetaan aliparametrit.

2.7.2 Haetaan niille arvo samoilla yo. periaatteilla kuin tavallisillekin parametreille. Niille ei kuitenkaan anneta lopullista totuusarvoa kuten yllä useimmiten viimeisessä askeleessa, vaan haetaan ainoastaan varsinainen arvo.

2.7.3 Verrataan saatuja arvoja annetulla operaatiolla ($<$, $\leq$, $>$, $\geq$, $=$ ja $\neq$), soveltaen sumeaa versiota ko. operaatiosta. Vertailuoperaatiot ovat analogisia yo. ala- ja ylärajavertailujen ja pistearvovertailujen kanssa.

2.8 Talletetaan laukeamistieto: lopputulos ja painoarvo. Jokaiselle parametrille tulee oma tietonsa, mutta koko säännön laukeamistiedot tulevat vasta nollatarkastuksien ja suodattimien vaikutusten laskennan jälkeen.

3 Nollatason tarkastus: Sellaiset arvot, jotka ovat alle säännössä määritellyn nollatason, asetetaan nollaksi. Mikäli säännölle ei ole asetettu omaa nollatasoa, verrataan yleiseen nollatasoon. Yleistä asetusta ei käytetä mikäli oma on asetettu, jotta voidaan saada myös yleistä nollatasoa alempia arvoja haluttaessa.

4 Suodattimien soveltaminen

4.1 Haetaan kaikki suodattimet ko. säännölle.

4.2 Järjestetään suodattimet numeroituun järjestykseen.

4.3 Sovelletaan suodattimia yksi kerrallaan, käyttäen aina edellisestä saatua tulosta alkuarvona seuraavalle.

4.3.1 Haetaan suodatinta vastaava suodatintyyppi

4.3.2 Sovelletaan suodatintyyppiä. Eri suodatintyypit eivät toiminnallisesti eroa juurikaan toisistaan, ainoastaan laskennallisesti, joten niitä ei esitellä tarkemmin tässä.

5 Talletetaan laukeamistieto. Tieto talletetaan ainoastaan kerran kaikkien suodattimien jälkeen, ei jokaiselle erikseen. Tässä on koko säännön laukeamistiedot. Mikäli säännöllä on alisääntöjä, niihin viitataan mutta tietoja ei kopioida enää uudestaan.

2.2 Paikkatiedonhallinnan algoritmi

1 Haetaan säännön argumentit

2 Tarkastetaan argumenttien oikea tyyppi, mikäli säännön argumentteja löytyi.

3 Avataan mahdolliset tiedostot

4 Tutkitaan säännön tyyppiä:

4.1 Kahden kohteen lähekkäisyys maastossa:

4.1.1 Haetaan kaikki reitit ruutujaon mukaan joilla kuljettujen ruutujen lukumäärä on pienin mahdollinen.

4.1.2 Haetaan kullekin ruudulle kulkukelpoisuus avatusta tiedostosta

4.1.3 Käydään läpi kaikki mahdolliset polut haetuilla ruuduilla ja lasketaan niistä kumulatiivinen kulkukelpoisuus.

4.1.4 Otetaan paras kyseisistä arvoista

4.2 Kahden kohteen lähekkäisyys teitä pitkin:

4.2.1 Haetaan kaikki reitit teitä pitkin avatusta tiedostosta, erottaen reiteistä erikseen erityyppiset tiet. Reitit jotka ylittävät tietyn pituuden hylätään automaattisesti. Tämä on helpoin tapa rajata haku äärelliseksi.
4.2.2 Lasketaan kullekin reitille tieosuuksien kulkukelpoisuuksien yhteistulos.
4.2.3 Otetaan paras kyseisistä arvoista.
4.3 Kahden kohteen sijainti samalla alueella, alue tunnettu
4.4 Kahden kohteen sijainti samalla alueella, alue tuntematon:
4.4.1 Haetaan ensimmäisen kohteen maastotyypi avatusta tiedostosta
4.4.2 Haetaan toisen kohteen maastotyypi avatusta tiedostosta
4.4.3 Haetaan kaikki maastoruutupolut kohteiden välillä joiden pituus on korkeintaan kaksi kertaa lyhimmän reitin pituus.
4.4.4 Haetaan kaikille polkujen ruuduille maastotyypit
4.4.5 Lopputulos saadaan vertaamalla maastotyyppejä ja hakemalla parhaiten niiden maastotyyppejä vastaava polku niiden väliltä ja soveltamalla and-operaattoria näihin tuloksiin.
4.5 Kahden kohteen sijainti samalla linjalla
4.6 Kahden kohteen lähekkäisyys
4.7 Kohde on ympäristötekijän vaikutusalueella
4.7.1 Haetaan kaikki ympäristötekijän rivit säännön taulu-kentän perusteella.
4.7.2 Mikäli ympäristötekijä on piste, verrataan etäisyyttä annettuun vaikutussäteeseen.
4.7.3 Mikäli ympäristötekijä on alue, tutkitaan vaikutusalueella olemista säännön R20 mukaan.
4.7.4 Lopputulos saadaan soveltamalla or-operaattoria kaikkiin saatuihin tuloksiin.
4.8 Kohteen ympäristö on tietyn tyyppinen
4.9 Kohde on solmupisteessä
4.9.1 Haetaan geometrisella haulla kohteesta lähtevät viivat
4.9.2 Kysytään callback-funktiolla ovatko ne särmäviivoja
4.9.3 Lasketaan särmien lukumäärä
4.9.4 Haetaan geometrisella haulla kohteessa olevat pisteet.
4.9.5 Kysytään callback-funktiolla ovatko ne solmupistekohteita
4.9.6 Lasketaan lukumäärä
4.9.7 Jos särmien lukumäärä = 1 (tarkasti, ei sumea) loppuarvo on 1.
4.9.8 Loppuarvo on (särmien lukumäärä > 2 or (särmien lukumäärä >= 2 and pisteiden lukumäärä > 2))
4.10 Kahden vierekkäisen solmun välinen särmä on lyhyt
4.11 Reitti on paras mahdollinen kahden solmun välillä
4.12 Verkon polku on kriittinen
4.12.1 Haetaan polku annettujen kohteiden välille
4.12.2 Käydään polku läpi solmupiste kerrallaan
4.12.3 Haetaan solmusta lähtevät viivat
4.12.4 Kysytään callback-funktiolla ovatko ne särmäviivoja
4.12.5 Lisätään särmien lukumäärä -2 aliverkkojen lukumäärään.
4.12.6 Kysytään callback-funktiolla lähtevän, polkuun kuulumattoman särmän kriittisyys. Tämä tieto haetaan kaikissa solmupisteissa ja kaikille särmille ja pidetään yllä tietoa mikä on ollut suurin arvo.
4.12.7 Kysytään callback-funktiolla kunkin polun päässä oleva asiakkaiden lukumäärä ja lisätään se summaan.

4.12.8 Lopputulos on polulta lähtevien alipolkujen lukumäärä > solmujen lukumäärä polulla) or suurin aliverkon kriittisyys or Asiakkaiden lukumäärien summa > annettu asiakasmäärä

4.12.9 Kun koko polku on käyty läpi, talletetaan polun pääsolmupisteisiin tiedot asiakkaiden määristä ja kriittisyydestä.[16]

4.13 Verkon polku on vakaa

4.14 Solmupiste on varmistettu

5 Talletetaan laukeamistieto

6 Nollatason tarkastus

7 Sovelletaan suodattimia

8 Talletetaan laukeamistieto

[16] Näin saadaan mahdollistettua myöhemmät kyselyt ja rekursiivinen käsittely koko verkolle laskematta kaikkea aina uudelleen heti tarvittaessa.